Optimizing short stabilizer codes for asymmetric channels

Alex Rigby,* J. C. Olivier, and Peter Jarvis
College of Sciences and Engineering, University of Tasmania, Hobart, Tasmania 7005, Australia

(Received 11 November 2019; accepted 4 February 2020; published 19 March 2020)

For a number of quantum channels of interest, phase-flip errors occur far more frequently than bit-flip errors. When transmitting across these asymmetric channels, the decoding error rate can be reduced by tailoring the code used to the channel. However, analyzing the performance of stabilizer codes on these channels is made difficult by the #P-completeness of optimal decoding. To address this, at least for short codes, we demonstrate that the decoding error rate can be approximated by considering only a fraction of the possible errors caused by the channel. Using this approximate error rate calculation, we extend a recent result to show that there are a number of $[[5 \leqslant n \leqslant 12, 1 \leqslant k \leqslant 3]]$ cyclic stabilizer codes that perform well on two different asymmetric channels. We also demonstrate that an indication of a stabilizer code's error rate is given by considering the error rate of a classical binary code related to the stabilizer. This classical error rate is far less complex to calculate, and we use it as the basis for a hill-climbing algorithm, which we show to be effective at optimizing codes for asymmetric channels. Furthermore, we demonstrate that simple modifications can be made to our hill-climbing algorithm to search for codes with desired structure requirements.

DOI: 10.1103/PhysRevA.101.032326

I. INTRODUCTION

Quantum codes can be employed to protect quantum information against the effects of a noisy channel. Of particular note are the stabilizer codes, which are defined by a stabilizer $\mathcal{S}$ that is an Abelian subgroup of the n-qubit Pauli group $\mathcal{P}_n$ [1]. An $[[n, k]]$ stabilizer code encodes the state of a k-qubit system in that of an n-qubit system; that is, it is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ of dimension 2^k. For a Pauli channel, an error E acting on the code is also an element of $\mathcal{P}_n$, with the component acting on any given qubit being I with probability p_I, X with probability p_X, Y with probability p_Y, or Z with probability p_Z. Most stabilizer codes are implicitly designed for good decoding performance (that is, a low decoding error rate) on the depolarizing channel, where $p_X = p_Y = p_Z$. This is achieved by ensuring that the code has large distance d, which is the weight of the lowest weight error that yields a trivial syndrome while having a nontrivial effect on the code. However, for a number of channels of physical interest, Z errors occur far more frequently than X errors [2,3]. For these channels, better decoding performance can be achieved by using codes that are tailored to the channel [4,5].

In this paper, our focus is on the construction of stabilizer codes for two different asymmetric channels. The first of these is the biased XZ channel, for which the X and Z components of an error occur independently at different rates. The second is a Pauli approximation of the combined amplitude damping (AD) and dephasing channel. Both of these channels have two degrees of freedom, which means that the values of p_X, p_Y, and p_Z can be defined via the total error probability $p = p_X + p_Y + p_Z$ and bias $\eta = p_Z/p_X$ [4,6]. A well-studied approach to constructing codes for asymmetric channels is to restrict consideration to Calderbank-Shor-Steane (CSS) codes [7,8], which can be designed to have separate X and Z distances d_X and d_Z (typically $d_Z > d_X$) [6,9–13]. We wish to take a more direct approach to the problem by actually determining the decoding error rates of the codes we construct (this also allows us to meaningfully consider non-CSS codes). However, to do this, we have to overcome the #P-completeness of stabilizer decoding [14], which stems from the equivalence of errors up to an element of the stabilizer. To achieve this, at least for short codes (that is, codes with small n), we first demonstrate that the error rate of an optimal decoder can be approximated by considering only a small subset $\mathcal{E}$ of the 4^n possible Pauli errors. We derive a bound on the relative error in this approximation, and we demonstrate that the independence of error components can be exploited to construct $\mathcal{E}$ without having to enumerate all possible errors. We also show that the performance of a classical $[2n, n + k]$ binary linear code associated with the stabilizer [1,15] gives an indication of the stabilizer code's performance (note that whenever we mention a code's performance or error rate, we are referring to that of the associated decoder). It is several orders of magnitude faster to calculate this classical error rate, and we show that it can itself be approximated using a limited error set.

We have a particular focus on cyclic codes, which are stabilizer codes based on classical self-orthogonal additive cyclic GF(4) codes [16–18] [where GF(q) is the q-element finite field]. This is motivated by the recent result of Ref. [4], where it was shown that a [[7,1]] cyclic code performs near optimally compared to 10 000 randomly constructed codes on the biased XZ channel for a range of error probabilities and

*alex.rigby@utas.edu.au

biases. We extend this result by enumerating the $[[5 \leqslant n \leqslant 12, 1 \leqslant k \leqslant 3]]$ cyclic codes and making use of our approximate error rate calculation. In particular, we demonstrate that there are a number of cyclic codes that perform well compared to the best of 10 000 randomly constructed codes for both the biased XZ and AD channels across a range of p and η values. In some cases, such as $[[n \geqslant 9, 1]]$ codes for the biased XZ channel, the best cyclic codes significantly outperform the best of the random codes constructed. To improve on the poor performance of the random search, we demonstrate the effectiveness of a simple hill-climbing algorithm that attempts to optimize the performance of the classical binary code associated with a stabilizer. We also show that by modifying the mutation operation employed by this hill-climbing algorithm, we can effectively search for codes with desired structure. In particular, we show that we can search for codes with weight-four generators, CSS codes, and linear codes.

The paper is organized as follows. Section II gives an overview of classical codes, asymmetric quantum channels, and stabilizer codes. In Sec. III, we detail our methods for calculating approximate error rates. In Sec. IV, we demonstrate the performance of cyclic codes, outline our hill-climbing search algorithm, and show its effectiveness. The paper is concluded in Sec. V.

II. BACKGROUND

A. Classical codes

A classical channel Φ maps a set of inputs $\mathcal{A}_x$ to a set of outputs $\mathcal{A}_y$. We are interested in the case where $\mathcal{A}_x = \mathcal{A}_y = \mathrm{GF}(q)$, for which the action of the channel is given by

$$\Phi(x) = x + e = y, \tag{1}$$

where $x \in \mathrm{GF}(q)$ is the channel input, $y \in \mathrm{GF}(q)$ is the channel output, and $e \in \mathrm{GF}(q)$ is an error (or noise) symbol that occurs with probability $P(e)$. Φ is called symmetric if $P(0) = 1 - p$ and $P(e_i) = p/(q-1)$ for $e_i \neq 0$. The noise introduced by the channel can be protected against using a code $\mathcal{C} \subseteq \mathrm{GF}(q)^n$, whose elements are called codewords. The effect of the combined channel Φ^n, which is composed of n copies of Φ, on some codeword $\boldsymbol{x} \in \mathcal{C}$ is

$$\Phi^n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}, \tag{2}$$

where $\boldsymbol{y} \in \mathrm{GF}(q)^n$ is the channel output and $\boldsymbol{e} \in \mathrm{GF}(q)^n$ is an error "vector." Assuming that error components occur independently, the probability of $\boldsymbol{e} = (e_1, \ldots, e_n)$ occurring is

$$P(\boldsymbol{e}) = \prod_{i=1}^{n} P(e_i), \tag{3}$$

where $P(e_i)$ is the probability of the error symbol e_i occurring on Φ. It follows that for a symmetric channel, the probability of an error $\boldsymbol{e}$ occurring depends only on its weight $w(\boldsymbol{e})$, which is the number of nonzero components from which it is composed. The distance d of a code is the weight of the lowest weight error mapping one codeword to another. The (minimum) weight $w(\mathcal{C})$ of a code $\mathcal{C}$ is simply the weight of the lowest weight codeword it contains.

A code is called additive if it forms a group (under addition) and linear if it forms a vector space. Such codes can be described by a generator matrix

$$G^T = (\boldsymbol{b}_1 \cdots \boldsymbol{b}_k), \tag{4}$$

where $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}$ is either a generating set or basis, respectively (note that we consider codewords as column vectors). A linear code can also be defined as the kernel of a $\mathrm{GF}(q)$ parity-check matrix H; that is,

$$\mathcal{C} = \{\boldsymbol{x} \in \mathrm{GF}(q)^n : H\boldsymbol{x} = \boldsymbol{0}\}. \tag{5}$$

If H has m rows, then $\dim(\mathcal{C}) = k \geqslant n - m$, with equality when H is full rank. For a linear code, the errors mapping one codeword to another are themselves codewords; therefore, it follows that the distance of a linear code $\mathcal{C}$ is simply $d = w(\mathcal{C})$. A linear code of length n with dimension k and distance d is called an $[n, k]_q$ or $[n, k, d]_q$ code (the q is typically omitted for binary codes, where $q = 2$). More generally, a length-n code of size $|\mathcal{C}| = K$ and distance d is called an $(n, K)_q$ or $(n, K, d)_q$ code.

The dual code of some $\mathcal{C} \subseteq \mathrm{GF}(q)^n$ with respect to the inner product $\langle \cdot, \cdot \rangle : \mathrm{GF}(q)^n \times \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ is

$$\mathcal{C}^\perp = \{\boldsymbol{c} \in \mathrm{GF}(q)^n : \langle \boldsymbol{c}, \boldsymbol{x} \rangle = 0 \ \forall \ \boldsymbol{x} \in \mathcal{C}\}. \tag{6}$$

$\mathcal{C}^\perp$ is the annihilator of $\mathcal{C}$ and is therefore a linear code. If $\mathcal{C}^\perp \subseteq \mathcal{C}$, then $\mathcal{C}$ is called dual containing; if $\mathcal{C} \subseteq \mathcal{C}^\perp$, then $\mathcal{C}$ is called self-orthogonal; and if $\mathcal{C}^\perp = \mathcal{C}$, then $\mathcal{C}$ is called self-dual. Note that if $\mathcal{C}$ is dual containing, then $\mathcal{C}^\perp$ is self-orthogonal and vice versa. Unless otherwise specified, the dual code is with respect to the Euclidean inner product

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle = \boldsymbol{c} \cdot \boldsymbol{x} = \sum_{i=1}^{n} c_i x_i. \tag{7}$$

In this case, if $\mathcal{C}$ is linear with generator matrix G, then a necessary and sufficient condition for $\boldsymbol{c} \in \mathcal{C}^\perp$ is $G\boldsymbol{c} = \boldsymbol{0}$; that is, a generator matrix for $\mathcal{C}$ is a parity-check matrix for $\mathcal{C}^\perp$. Conversely, if H is a parity-check matrix for $\mathcal{C}$, then it is a generator matrix for $\mathcal{C}^\perp$.

A decoder uses the output of a channel to infer its input. For a linear code, this inference can be aided by the syndrome

$$\boldsymbol{z} = H\boldsymbol{y} = H(\boldsymbol{x} + \boldsymbol{e}) = H\boldsymbol{e}. \tag{8}$$

As channel outputs that differ only by a codeword yield the same syndrome, the q^{n-k} possible syndromes can be associated with the cosets of $\mathrm{GF}(q)^n/\mathcal{C}$. Given some syndrome measurement $\boldsymbol{z}$, an optimal maximum *a posteriori* (MAP) decoder will then return the most probable error

$$\hat{\boldsymbol{e}}_{\boldsymbol{z}} = \underset{\boldsymbol{e} \in \mathrm{GF}(q)^n}{\mathrm{argmax}} P(\boldsymbol{e}|\boldsymbol{z}) \tag{9}$$

in the corresponding coset. The channel input can then be inferred as $\hat{\boldsymbol{x}} = \boldsymbol{y} - \hat{\boldsymbol{e}}_{\boldsymbol{z}}$. If $\hat{\boldsymbol{e}} = \hat{\boldsymbol{e}}_{\boldsymbol{z}}$ (and hence $\hat{\boldsymbol{x}} = \boldsymbol{x}$), then decoding is successful; otherwise, a decoding error has occurred. The probability of such a decoding error, called the frame error rate (FER), is simply

$$F = 1 - \sum_{\boldsymbol{z} \in \mathrm{GF}(q)^{n-k}} P(\hat{\boldsymbol{e}}_{\boldsymbol{z}}). \tag{10}$$

Unfortunately, even in the simple case of a binary code operating on the binary symmetric channel (a symmetric channel with $q = 2$), this decoding problem can be shown to be NP-complete [19]. This complicates the design of highly performant codes (that is, codes yielding a low FER). In practice, when designing codes for symmetric channels, the simpler goal of achieving a large distance is typically settled for. This is motivated by the fact that for low-distance codes, there are many errors in each coset $\hat{\boldsymbol{e}}_z + \mathcal{C}$ with weight, and hence probability, similar to $\hat{\boldsymbol{e}}_z$, which leads to a high FER according to Eq. (10) (see Sec. II A of Ref. [20] for a more detailed discussion).

Two codes $\mathcal{C}$ and $\mathcal{C}'$ are called permutation equivalent if they are the same up to a relabeling of coordinates. Permutation-equivalent codes share a large number of properties including length, size, and distance; furthermore, they yield the same FER for channels where the error components are independently and identically distributed. While there are more general notions of code equivalence, whenever we say that two codes are equivalent, we mean that they are permutation equivalent in this paper. Furthermore, if some family (set) of codes $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ can be split into M equivalence classes (according to permutation equivalence), then we simply say that M of the codes are inequivalent.

B. Cyclic codes

Cyclic codes are those for which a cyclic shift of any codeword is also a codeword; that is, for a cyclic code $\mathcal{C}$, if $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$, then it is also the case that $(c_{n-1}, c_0, \ldots, c_{n-2}) \in \mathcal{C}$ (note that to be consistent with standard convention, we index the codewords of cyclic codes from zero in this section). If $\mathcal{C}$ is linear, then it has a convenient description through the mapping

$$\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) \leftrightarrow c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} = c(x) \tag{11}$$

of codewords to polynomials in GF$(q)[x]$. Cyclic shifts of codewords correspond to a multiplication by x taken modulo $x^n - 1$; that is, $(c_{n-1}, c_0, \ldots, c_{n-2}) \leftrightarrow xc(x) \,(\mathrm{mod}\, x^n - 1)$. As $\mathcal{C}$ is linear, $r(x)c(x) \,(\mathrm{mod}\, x^n - 1)$ is a codeword for any $r(x) \in$ GF$(q)[x]$, from which it follows that $\mathcal{C}$ corresponds to an ideal $I_{\mathcal{C}} \in$ GF$(q)[x]/(x^n - 1)$. Any such ideal is principal and is generated by a unique monic polynomial of minimal degree $g(x) \in I_{\mathcal{C}}$ that is a factor of $x^n - 1$ [21]; through slight abuse of notation, we write $\mathcal{C} = \langle g(x) \rangle$. $\mathcal{C}$ has dimension $k = n -$ deg(g) and has a generator matrix

$$G = \begin{pmatrix} g_0 & \cdots & g_{n-k} & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & g_0 & \cdots & g_{n-k} \end{pmatrix}. \tag{12}$$

Furthermore, a parity-check matrix

$$H = \begin{pmatrix} h_k & \cdots & h_0 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & h_k & \cdots & h_0 \end{pmatrix} \tag{13}$$

is given in terms of the check polynomial $h(x) = (x^n - 1)/g(x)$. It follows that the dual code $\mathcal{C}^{\perp}$ is also cyclic and is generated by $x^k h(x^{-1})$.

In the quantum setting, we are particularly interested in codes over GF$(4) = \{0, 1, \omega, \omega^2 = \bar{\omega}\}$ that are self-orthogonal with respect to the trace inner product (this will be explained further in Sec. II D). Note that the trace inner product of $\boldsymbol{a}, \boldsymbol{b} \in$ GF$(4)^n$ is

$$\boldsymbol{a} * \boldsymbol{b} = \mathrm{tr}(\boldsymbol{a} \cdot \bar{\boldsymbol{b}}) = \mathrm{tr}\left(\sum_{i=1}^{n} a_i \bar{b}_i \right), \tag{14}$$

where $\bar{0} = 0$, $\bar{1} = 1$, $\bar{\omega} = \omega^2$, and $\bar{\omega^2} = \omega$; and $\mathrm{tr}(x) = x + \bar{x}$ [that is, $\mathrm{tr}(0) = \mathrm{tr}(1) = 0$ and $\mathrm{tr}(\omega) = \mathrm{tr}(\bar{\omega}) = 1$]. A linear cyclic GF(4) code $\mathcal{C} = \langle g(x) \rangle$ is self-orthogonal if and only if $g(x)g^{\dagger}(x) \equiv 0 \,(\mathrm{mod}\, x^n - 1)$ [16], where

$$g^{\dagger}(x) = \bar{g}_0 + \sum_{j=1}^{n-1} \bar{g}_{n-j} x^j. \tag{15}$$

More generally, an $(n, 2^k)_4$ additive cyclic code $\mathcal{C}$ has two generators [16–18]. Following the formulation of Ref. [16], $\mathcal{C} = \langle \omega p(x) + q(x), r(x) \rangle$ where $p(x), q(x), r(x) \in$ GF$(2)[x]$; $p(x)$ and $r(x)$ are factors of $x^n - 1$; and $r(x)$ is also a factor of $q(x)(x^n - 1)/p(x)$. In general, the choice of generators is not unique; however, any other representation will be of the form $\mathcal{C} = \langle \omega p(x) + q'(x), r(x) \rangle$ where $q'(x) \equiv q(x) \,[\mathrm{mod}\, r(x)]$. The size of $\mathcal{C}$ is given by $k = 2n - \deg(p) - \deg(r)$, with a generator matrix consisting of $n - \deg(p)$ cyclic shifts of the codeword corresponding to $\omega p(x) + q(x)$ and $n - \deg(r)$ cyclic shifts of the codeword corresponding to $r(x)$. $\mathcal{C}$ is self-orthogonal (with respect to the trace inner product) if and only if

$$p(x)r(x^{n-1}) \equiv p(x^{n-1})r(x) \equiv 0 \,(\mathrm{mod}\, x^n - 1), \tag{16}$$

$$p(x)q(x^{n-1})r(x) \equiv p(x^{n-1})q(x) \,(\mathrm{mod}\, x^n - 1). \tag{17}$$

It is possible to enumerate all the self-orthogonal $(n, 2^k)_4$ additive cyclic codes through a slight modification of the method presented in Ref. [22]: $r(x)$ ranges over all factors of $x^n - 1$; for each $r(x)$, $p(x)$ ranges over the factors of $x^n - 1$ of degree $2n - k - \deg(r)$ that satisfy Eq. (16); and for each pair of $r(x)$ and $p(x)$, $q(x)$ ranges over the polynomials with $\deg(q) \leqslant \deg(r)$ that satisfy both Eq. (17) and $q(x)(x^n - 1) \equiv 0 \,[\mathrm{mod}\, p(x)r(x)]$.

While every additive cyclic code has a "canonical" representation involving two generators, many of them can be described using only one [17,18] (that is, they have a generating set composed of cyclic shifts of a single codeword). This is guaranteed to be the case if $r(x) = x^n - 1$ or if $p(x) = x^n - 1$ and $q(x)$ is a multiple of $r(x)$. However, these are not necessary conditions for a single-generator representation to exist. For example, there is a $(5, 2^5)_4$ code with $p(x) = 1 + x$, $q(x) = x^3$, and $r(x) = 1 + x + x^2 + x^3$, which gives a canonical generator matrix

$$G = \begin{pmatrix} \omega & \omega & 0 & 1 & 0 \\ 0 & \omega & \omega & 0 & 1 \\ 1 & 0 & \omega & \omega & 0 \\ 0 & 1 & 0 & \omega & \omega \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}; \tag{18}$$

however, it is also has the generator matrix

$$G' = \begin{pmatrix} \omega & \omega & 0 & 1 & 0 \\ 0 & \omega & \omega & 0 & 1 \\ 1 & 0 & \omega & \omega & 0 \\ 0 & 1 & 0 & \omega & \omega \\ \omega & 0 & 1 & 0 & \omega \end{pmatrix}. \tag{19}$$

We can express this code compactly as $\mathcal{C} = \langle \omega\omega 010, 11111 \rangle_{\text{cyc}} \equiv \langle \omega\omega 010 \rangle_{\text{cyc}}$.

C. Quantum channels

The action of a quantum channel Φ on a quantum state described by the density operator ρ is

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{20}$$

where the A_k, called Kraus operators, satisfy $\sum_k A_k^\dagger A_k = I$ (the identity operator) [23]. We are interested in qubit systems, for which states belong to a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. Furthermore, we are concerned with Pauli channels, which are of the form

$$\Phi(\rho) = p_I \rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z, \tag{21}$$

where $p_I + p_X + p_Y + p_Z = 1$, and in the computational $\{|0\rangle, |1\rangle\}$ basis

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{22}$$

The action of this channel can be interpreted as mapping a pure state $|\phi\rangle$ to $E|\phi\rangle$ where the error E is I with probability p_I, X with probability p_X, Y with probability p_Y, or Z with probability p_Z [24]. X can be viewed as a bit-flip operator as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Z can be viewed as a phase flip as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. $Y = iXZ$ can be viewed as a combined bit and phase flip.

The quantum equivalent of the symmetric channel is the depolarizing channel, for which $p_I = 1 - p$ and $p_X = p_Y = p_Z = p/3$. For a number of systems of physical interest, phase-flip errors occur far more frequently than bit-flip errors [2,3]. We focus on two such asymmetric channels in this paper. The first is the biased XZ channel, for which the X and Z components of an error $E \propto X^u Z^v$, where $u, v \in \text{GF}(2)$, occur independently with probabilities q_X and q_Z, respectively. It follows from the independence of the error components that $p_X = q_X(1 - q_Z)$, $p_Z = q_Z(1 - q_X)$, and $p_Y = q_X q_Z$. A typical way to specify an asymmetric channel with two degrees of freedom is through the total error probability $p = p_X + p_Y + p_Z$ and bias $\eta = p_Z/p_X$. Note that while this definition of bias is consistent with Refs. [4,6], some authors give alternate definitions; for example, bias is defined as $p_Z/(p_X + p_Y)$ in Ref. [5] and $(p_Y + p_Z)/(p_X + p_Y)$ in Ref. [25]. Ultimately, the exact nature of the channel parametrization will have no real impact on our results, which has lead us to select the simplest definition of bias. The second channel of interest is the combined amplitude damping (AD) and dephasing channel, which is described by the non-Pauli Kraus operators

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda-\gamma} \end{pmatrix}, \tag{23}$$

$$A_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}, \tag{24}$$

$$A_2 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{pmatrix}. \tag{25}$$

A Pauli approximation of this channel can be obtained through a process called Pauli twirling [26–28]. In particular, the approximate channel is [6]

$$\Phi_T(\rho) = \frac{1}{4} \sum_{\sigma \in \{I,X,Y,Z\}} \sigma^\dagger \Phi(\sigma\rho\sigma^\dagger)\sigma \tag{26}$$

$$= \frac{2 - \gamma + 2\sqrt{1-\lambda-\gamma}}{4}\rho + \frac{\gamma}{4} X\rho X + \frac{\gamma}{4} Y\rho Y + \frac{2 - \gamma - 2\sqrt{1-\lambda-\gamma}}{4} Z\rho Z. \tag{27}$$

Again, this channel has two degrees of freedom (λ and γ) and can therefore be described in terms of the total error probability p and bias $\eta = p_Z/p_X$. Note that in the case of $\eta = 1$, Φ_T reduces to the depolarizing channel. For the sake of brevity, we will simply refer to Φ_T as the AD channel.

The Pauli matrices are Hermitian, unitary, and anticommute with each other. Furthermore, they form a group

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z \rangle \tag{28}$$

called the Pauli group. The n-qubit Pauli group $\mathcal{P}_n$ is composed of all n-fold tensor product combinations of elements of $\mathcal{P}_1$. Note that when writing elements of $\mathcal{P}_n$, the tensor products are often implied; for example, we may write $I \otimes I \otimes X \otimes I \otimes Y \otimes Z \otimes I \otimes I \in \mathcal{P}_8$ as $IIXIYZII$. The weight $w(g)$ of some $g \in \mathcal{P}_n$ is the number of nonidentity components from which it is composed. It follows from the commutation relations of the Pauli matrices that any two elements of $\mathcal{P}_n$ commute if their nonidentity components differ in an even number of places; otherwise, they anticommute.

As in the classical case, the noise introduced by a quantum channel can be protected against using a code. In the qubit case, a code is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ whose elements are again called codewords. These codewords are transmitted across the combined n-qubit channel $\Phi^{\otimes n}$, which, in the Pauli case, maps a codeword $|\phi\rangle$ to $E|\phi\rangle$ where $E \in \mathcal{P}_n$. Similar to the classical case of Eq. (3), if the error components are independent, then the probability of an error $E = E_1 \otimes \cdots \otimes E_n$ occurring (up to phase) is

$$P(E) = \prod_{i=1}^{n} P(E_i), \tag{29}$$

where $P(E_i)$ is the probability of the error E_i occurring (up to phase) on the single-qubit channel Φ. The equivalence of errors up to phase can be addressed more explicitly by instead considering $\tilde{E} = \{E, -E, iE, -iE\} \in \mathcal{P}_n/\{\pm I, \pm iI\} = \tilde{\mathcal{P}}_n$.

D. Stabilizer codes

Stabilizer codes are defined by an Abelian subgroup $\mathcal{S} < \mathcal{P}_n$, called the stabilizer, that does not contain $-I$ [1]. The code $\mathcal{Q}$ is the space of states that are fixed by every element $s_i \in \mathcal{S}$; that is,

$$\mathcal{Q} = \{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} : s_i|\phi\rangle = |\phi\rangle \, \forall \, s_i \in \mathcal{S}\}. \tag{30}$$

The requirement that $-I \notin \mathcal{S}$ means both that no $s \in \mathcal{S}$ can have a phase factor of $\pm i$, and also that if $s \in \mathcal{S}$, then $-s \notin \mathcal{S}$. If $\mathcal{S}$ is generated by $M = \{M_1, \ldots, M_m\} \subset \mathcal{P}_n$, then it is sufficient (and obviously necessary) for $\mathcal{Q}$ to be stabilized by every M_i. Assuming that the set of generators is minimal, which will be the case for all codes considered in this paper, it can be shown that $\dim(\mathcal{Q}) = 2^k$ where $k = n - m$ [24]; that is, $\mathcal{Q}$ encodes the state of a k-qubit system.

Suppose an error E occurs, mapping some codeword $|\phi\rangle \in \mathcal{Q}$ to $E|\phi\rangle$. A projective measurement of a generator M_i will give the result $+1$ if $[E, M_i] = EM_i - M_iE = 0$ or -1 if $\{E, M_i\} = EM_i + M_iE = 0$. These measurement values define the syndrome $\boldsymbol{z} \in \mathrm{GF}(2)^{n-k}$ with

$$z_i = \begin{cases} 0 & \text{if } [E, M_i] = 0, \\ 1 & \text{if } \{E, M_i\} = 0. \end{cases} \tag{31}$$

Defining $\tilde{\mathcal{S}} = \{\tilde{s} = \{s, -s, is, -is\} : s \in \mathcal{S}\}$, the syndrome resulting from $\tilde{E} \in \tilde{\mathcal{P}}_n$ depends only on which coset of $\tilde{\mathcal{P}}_n/N(\tilde{\mathcal{S}})$ it belongs to, where $N(\tilde{\mathcal{S}}) = \{g \in \tilde{\mathcal{P}}_n : g^{-1}\tilde{\mathcal{P}}_n g = \tilde{\mathcal{P}}_n\}$ is the normalizer of $\tilde{\mathcal{S}}$ in $\tilde{\mathcal{P}}_n$; furthermore, the effect of $\tilde{E}$ on the code depends only on which coset of $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$ it belongs to [1]. Note that as $\tilde{\mathcal{S}} \lhd N(\tilde{\mathcal{S}})$, the 2^{n-k} cosets of $\tilde{\mathcal{P}}_n/N(\tilde{\mathcal{S}})$ are each the union of 2^{2k} cosets of $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$. In the classical case, the distance d of a linear code is equal to the weight of the lowest weight error yielding a trivial syndrome while having a nontrivial effect on the code. This extends to the quantum case, with the distance d of a stabilizer code being the weight of the lowest weight element in $N(\tilde{\mathcal{S}})\backslash\tilde{\mathcal{S}}$ [1]. An n-qubit code of dimension 2^k and distance d is called an $[[n, k]]$ or $[[n, k, d]]$ code (the double brackets differentiate it from a classical code).

Given the equivalence of errors up to an element of the stabilizer, a MAP decoder will determine the most likely coset

$$\hat{A}_{\boldsymbol{z}} = \underset{A \in \tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}}{\operatorname{argmax}} P(A|\boldsymbol{z}) \tag{32}$$

that is consistent with the syndrome measurement. If $\hat{A}_{\boldsymbol{z}}$ has the representative $\hat{\tilde{E}} = \{\hat{E}, -\hat{E}, i\hat{E}, -i\hat{E}\}$, then the decoder attempts correction by applying $\hat{E}$ to the channel output. If $\tilde{E} \in \hat{A}_{\boldsymbol{z}}$, and hence $\hat{\tilde{E}}\tilde{E} \in \tilde{\mathcal{S}}$, then decoding is successful; otherwise, a decoding error has occurred. It therefore follows that the FER is

$$F_{\mathrm{MAP}} = 1 - \sum_{\boldsymbol{z} \in \mathrm{GF}(2)^{n-k}} P(\hat{A}_{\boldsymbol{z}}). \tag{33}$$

Unfortunately, this decoding problem has been shown to be #P-complete [14]. Furthermore, the simpler decoding problem of determining the single most likely error

$$\hat{\tilde{E}}_{\boldsymbol{z}} = \underset{\tilde{E} \in \tilde{\mathcal{P}}_n}{\operatorname{argmax}} P(\tilde{E}|\boldsymbol{z}) \tag{34}$$

corresponding to the observed syndrome is essentially the same as the classical decoding problem outlined in Sec. II A and hence is also NP-complete [29–31]. The FER for this decoder is

$$F_{\mathrm{MAP-SE}} = 1 - \sum_{\boldsymbol{z} \in \mathrm{GF}(2)^{n-k}} P(\hat{\tilde{E}}_{\boldsymbol{z}}\tilde{\mathcal{S}}), \tag{35}$$

where "SE" stands for "single error."

Two stabilizers (or the codes they define) are permutation equivalent if they are equal up to a relabeling of qubits. As in the classical case, if two stabilizer codes are permutation equivalent, then they are both $[[n, k, d]]$ codes; furthermore, they will yield the same FERs (both F_{MAP} and $F_{\mathrm{MAP-SE}}$) when the error components are independently and identically distributed, which is the case for the channels that we consider. Again, while there are more general notions of quantum code equivalence, we are always referring to permutation equivalence in this paper.

The links between stabilizer codes and classical codes can be made more concrete by representing the elements of $\tilde{\mathcal{P}}_n$ as elements of $\mathrm{GF}(2)^{2n}$ [1,15]. This is achieved via the isomorphism

$$X^{u_1}Z^{v_1} \otimes \cdots \otimes X^{u_n}Z^{v_n} = X^{\boldsymbol{u}}Z^{\boldsymbol{v}} \leftrightarrow (\boldsymbol{u}|\boldsymbol{v}), \tag{36}$$

with the product of elements in $\tilde{\mathcal{P}}_n$ corresponding to addition in $\mathrm{GF}(2)^{2n}$. Furthermore, representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the symplectic inner product of the binary representations is zero, where the symplectic inner product of $\boldsymbol{a} = (\boldsymbol{u}|\boldsymbol{v}), \boldsymbol{b} = (\boldsymbol{u}'|\boldsymbol{v}') \in \mathrm{GF}(2)^{2n}$ is $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v}$. Utilizing this isomorphism, the generators of some stabilizer $\mathcal{S}$ can be used to define the rows of an $m \times 2n$ binary matrix

$$H = (H_X|H_Z), \tag{37}$$

where H_X and H_Z are each $m \times n$ matrices. Under this mapping, the requirement that all stabilizer generators commute becomes

$$H_X H_Z^T + H_Z H_X^T = 0. \tag{38}$$

Conversely, a $[2n, n + k]$ linear binary code $\mathcal{C}$ with a parity-check matrix H satisfying this constraint can be used to define a stabilizer $\mathcal{S}$. Technically, this only specifies $\tilde{\mathcal{S}}$; however, as previously outlined, it is $\tilde{\mathcal{S}}$ that dictates the effect of an error on a stabilizer code, which means that the 2^{n-k} stabilizers corresponding to $\tilde{\mathcal{S}}$ will all have the same error correction properties (the codes corresponding to each such stabilizer actually form a partition of $(\mathbb{C}^2)^{\otimes n}$ [32,33]). Without loss of generality, we can therefore map $\tilde{\mathcal{S}}$ to a particular stabilizer $\mathcal{S}$ by arbitrarily selecting a phase factor of $+1$ for all the generators.

A subclass of stabilizer codes are the Calderbank-Shor-Steane (CSS) codes [7,8], which have a binary representation of the form

$$H = \left(\begin{array}{c|c} \tilde{H}_X & 0 \\ 0 & \tilde{H}_Z \end{array}\right). \tag{39}$$

For such codes, the commutation condition of Eq. (38) becomes $\tilde{H}_Z\tilde{H}_X^T = 0$, which is satisfied when $\mathcal{C}_X^{\perp} \subseteq \mathcal{C}_Z$, where $\mathcal{C}_X$ and $\mathcal{C}_Z$ are classical codes defined by the parity-check matrices $\tilde{H}_X$ and $\tilde{H}_Z$, respectively. If $\mathcal{C}_X = \mathcal{C}_Z$, then this reduces to $\mathcal{C}_X^{\perp} \subseteq \mathcal{C}_X$, in which case, the CSS code is called dual containing (DC).

As previously mentioned, the decoding problem of Eq. (34) is essentially the same as the classical decoding problem. This link can be made more explicit by expressing errors within the binary framework using the mapping $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z} \leftrightarrow \boldsymbol{e} = (\boldsymbol{e}_X^T | \boldsymbol{e}_Z^T)^T$ (where $\boldsymbol{e}_X$, $\boldsymbol{e}_Z$, and $\boldsymbol{e}$ are column vectors for consistency with the classical case). If the generators of a stabilizer define the parity-check matrix H for the binary code $\mathcal{C}$, then the syndrome corresponding to E can be found by taking the symplectic inner product of $\boldsymbol{e}$ with each row of H, which can be written compactly as

$$\boldsymbol{z} = H \begin{pmatrix} \boldsymbol{e}_Z \\ \boldsymbol{e}_X \end{pmatrix} = HP\boldsymbol{e}, \tag{40}$$

where

$$P = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}. \tag{41}$$

With this slight modification to classical syndrome calculation, determining $\tilde{\hat{E}}_z$ in Eq. (38) corresponds precisely to determining $\hat{\boldsymbol{e}}_z$ in Eq. (9). Note that some authors avoid this difference in syndrome calculation by using the mapping $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z} \leftrightarrow \boldsymbol{e} = (\boldsymbol{e}_Z^T | \boldsymbol{e}_X^T)^T$ [34], which gives $\boldsymbol{z} = H\boldsymbol{e}$ as in the classical case of Eq. (8). For a CSS code, the syndrome associated with an error $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$ is

$$\boldsymbol{z} = \begin{pmatrix} \tilde{H}_X \boldsymbol{e}_Z \\ \tilde{H}_Z \boldsymbol{e}_X \end{pmatrix} = \begin{pmatrix} \boldsymbol{z}_Z \\ \boldsymbol{z}_X \end{pmatrix}. \tag{42}$$

This allows the X and Z components of the error to be treated separately. In particular, $\boldsymbol{e}_Z$ can be inferred from $\tilde{H}_X \boldsymbol{e}_Z = \boldsymbol{z}_Z$, while $\boldsymbol{e}_X$ can be inferred from $\tilde{H}_Z \boldsymbol{e}_X = \boldsymbol{z}_X$. However, this approach is only guaranteed to determine the single most likely error if the X and Z components of E occur independently, which is the case for the biased XZ channel but not for the AD channel among others (see Sec. II E of Ref. [20] for a more detailed discussion).

Elements of $\tilde{\mathcal{P}}_n$ can also be represented as elements of $\mathrm{GF}(4)^n$ according to the isomorphism [1,16]

$$X^{\boldsymbol{u}} Z^{\boldsymbol{v}} \leftrightarrow \boldsymbol{u} + \omega \boldsymbol{v}, \tag{43}$$

with the product of elements in $\tilde{\mathcal{P}}_n$ corresponding to addition in $\mathrm{GF}(4)^n$. Representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the trace inner product [see Eq. (14)] of the corresponding elements of $\mathrm{GF}(4)^n$ is zero. Utilizing this isomorphism, any $(n, 2^{n-k})_4$ additive GF(4) code $\mathcal{C}$ that is self-orthogonal with respect to the trace inner product can be used to define an $[[n, k]]$ stabilizer code (it is for this reason that stabilizer codes are sometimes called additive codes). Furthermore, the generators of the stabilizer $\mathcal{S}$ can be associated with the rows of a generator matrix G for $\mathcal{C}$. We can describe a stabilizer code based on properties of $\mathcal{C}$; for example, if $\mathcal{C}$ is linear and/or cyclic, then we will also call $\mathcal{S}$ (and the code $\mathcal{Q}$ it defines) linear and/or cyclic.

Similar to the classical case, when designing a stabilizer code for the depolarizing channel, the complexity of determining its FER can be avoided by instead using code distance as something of a proxy. However, for asymmetric channels, distance becomes a less accurate metric as the probability of an error occurring no longer depends only on its weight. One approach in this case is to design codes with different X and Z distances, which are called $[[n, k, d_X/d_Z]]$ codes. For these so-called asymmetric codes, d_X and d_Z are the maximal values for which there is no $\tilde{E} \in N(\tilde{\mathcal{S}}) \backslash \tilde{\mathcal{S}}$ where $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$ and both $w(\boldsymbol{e}_X) < d_X$ and $w(\boldsymbol{e}_Z) < d_Z$. Such codes are typically constructed within the CSS framework, where $d_X = w(\mathcal{C}_Z \backslash \mathcal{C}_X^{\perp})$ and $d_X = w(\mathcal{C}_X \backslash \mathcal{C}_Z^{\perp})$ [35]. Outside of the CSS framework, where the X and Z components of an error cannot be considered separately, the distances d_X and d_Z are somewhat less meaningful and potentially not even unique. For example, the $(7, 2^6)_4$ additive cyclic code $\langle \omega 10 \omega 100 \rangle_{\mathrm{cyc}}$ maps to the [[7,1,3]] cyclic stabilizer code with $\mathcal{S} = \langle XZIZXII \rangle_{\mathrm{cyc}}$, which can be considered as a $[[7, 1, 7/1]]$, $[[7, 1, 1/7]]$, or $[[7, 1, 2/3]]$ code. Some examples of asymmetric codes (for qubits) can be found in Refs. [6,9–13].

III. APPROXIMATE FER CALCULATION

In this paper, we wish to construct stabilizer codes that perform well on asymmetric channels. In particular, we wish to gauge their performance directly; that is, we wish to accurately determine the FER exhibited by a MAP decoder as given in Eq. (33). As previously noted, determining this error rate is an #P-complete problem. In this section, we therefore investigate lower complexity methods of approximating F_{MAP} and derive bounds on the relative error of these approximations.

A. Limited error set

In most cases, many of the errors in $\tilde{\mathcal{P}}_n$ occur with very low probability. It seems reasonable to assume that ignoring these low-probability errors will have little effect on the FER calculation of Eq. (33). In particular, suppose we only consider a subset of errors $\mathcal{E} \subset \tilde{\mathcal{P}}_n$. We can calculate an approximate FER using $\mathcal{E}$ by first partitioning it by syndrome into the sets $B_1, \ldots, B_r$, where $r \leqslant 2^{n-k}$. Each of these B_i is then further partitioned by equivalence up to an element of $\tilde{\mathcal{S}}$ to give the sets $A_{i1}, \ldots, A_{is}$, where $s \leqslant 2^{2k}$. The approximate FER is then

$$F_{\mathcal{E}} = 1 - \sum_{i=1}^{r} \max_j P(A_{ij}) = 1 - \sum_{i=1}^{r} P(\hat{A}_i), \tag{44}$$

where

$$\hat{A}_i = \operatorname*{argmax}_{A_{ij} \in B_i} P(A_{ij}). \tag{45}$$

Note that if we wish to explicitly associate a stabilizer $\mathcal{S}$ with $F_{\mathcal{E}}$, then we write $F_{\mathcal{E}}^{\mathcal{S}}$. In the best case, $\mathcal{E}$ will contain every $\hat{A}_z$ in its entirety, which gives $\sum_z P(\hat{A}_z) = \sum_{i=1}^{r} P(\hat{A}_i)$ and hence $F_{\mathcal{E}} = F_{\mathrm{MAP}}$. In the worst case, $\sum_{i=1}^{r} P(\hat{A}_i) = \sum_z P(\hat{A}_z) - [1 - P(\mathcal{E})]$, which gives $F_{\mathcal{E}} = F_{\mathrm{MAP}} + (1 - P(\mathcal{E}))$. In general,

$$0 \leqslant F_{\mathcal{E}} - F_{\mathrm{MAP}} \leqslant 1 - P(\mathcal{E}), \tag{46}$$

which leads to

$$\begin{aligned} \delta_{\mathcal{E}} &= \frac{F_{\mathcal{E}} - F_{\mathrm{MAP}}}{F_{\mathrm{MAP}}} \\ &\leqslant \frac{1 - P(\mathcal{E})}{F_{\mathrm{MAP}}} \\ &\leqslant \frac{1 - P(\mathcal{E})}{F_{\mathcal{E}} - [1 - P(\mathcal{E})]} \qquad (47) \\ &= \Delta_{\mathcal{E}}. \qquad (48) \end{aligned}$$

This bound $\Delta_{\mathcal{E}}$ on the relative error $\delta_{\mathcal{E}}$ in the approximate FER calculation is of practical use as it does not require any knowledge of F_{MAP}.

There are two desirable attributes of the set $\mathcal{E} \subset \tilde{\mathcal{P}}_n$ used to calculate $F_{\mathcal{E}}$. The first of these, which follows from Eq. (47), is for $1 - P(\mathcal{E})$ to be less than some predetermined value as this affects the accuracy of $F_{\mathcal{E}}$. The second is for $|\mathcal{E}|$ to be small as this reduces the complexity of calculating $F_{\mathcal{E}}$. It is possible to construct such a set without enumerating $\tilde{\mathcal{P}}_n$ in its entirety by exploiting the independence of error components, which means that the probability of an error occurring depends only on the number of I, X, Y, and Z components it contains. Explicitly, the probability of some error $\tilde{E} \in \tilde{\mathcal{P}}_n$ occurring is

$$P(\tilde{E}) = \prod_{\sigma \in \{I,X,Y,Z\}} p_{\sigma}^{n(\sigma)}, \tag{49}$$

where $n(\sigma)$ is the number of tensor components of E that are equal to σ up to phase. Furthermore, the number of errors in $\tilde{\mathcal{P}}_n$ with a given distribution of components is [36]

$$N = \frac{n!}{n(I)!n(X)!n(Y)!n(Z)!}. \tag{50}$$

Therefore, to construct $\mathcal{E}$, we first enumerate all of the possible combinations of $n(I)$, $n(X)$, $n(Y)$, and $n(Z)$ such that $n(I) + n(X) + n(Y) + n(Z) = n$, which is a straightforward variation of the integer partition problem [37]. These combinations are sorted in descending order according to their associated probability as given in Eq. (49). In an iterative process, we then work through this list of combinations, adding the N distinct errors associated with each one to $\mathcal{E}$ until we reach the desired value of $1 - P(\mathcal{E})$. This construction has the added benefit of ensuring that $\mathcal{E}$ is permutation invariant, which guarantees that $F_{\mathcal{E}}$ will be the same for equivalent codes.

For the approximate error rate calculation presented in this section to be of any real use, it must be accurate even when $\mathcal{E}$ is relatively small. To demonstrate that this is in fact the case, we have first constructed 1 000 random [[7,1]] codes. To produce a random stabilizer $\mathcal{S} = \langle M_1, \ldots, M_{n-k} \rangle$, we iteratively select $\tilde{M}_i = \{M_i, -M_i, iM_i, -iM_i\}$ at random from from $N(\langle \tilde{M}_1, \ldots, \tilde{M}_{i-1} \rangle) \backslash \langle \tilde{M}_1, \ldots, \tilde{M}_{i-1} \rangle$ (note that we arbitrarily use a phase factor $+1$ for each M_i as outlined in Sec. II D). Our only structure constraint on $\mathcal{S}$ is that it must involve every qubit; that is, for all $1 \leqslant j \leqslant n$, there must be some $M_i^{(j)} \not\propto I$, where $M_i^{(j)}$ is the jth tensor component of M_i (if a stabilizer does not satisfy this constraint, we simply discard it and construct a new one). For biased XZ channels with $p = 0.1$, 0.01, or 0.001 and $\eta = 1$, 10, or 100, we have then determined the fraction of the 1 000 codes that yield a relative error $\delta_{\mathcal{E}} \leqslant 0.01$ or relative error bound $\Delta_{\mathcal{E}} \leqslant 0.01$ for varying $|\mathcal{E}|$. The results of this are shown in Fig. 1 where it can be seen that, depending on the channel parameters, only 1–5% of $\tilde{\mathcal{P}}_n$ needs to be considered to yield $\delta_{\mathcal{E}} \leqslant 0.01$ for every code. As is to be expected, a slightly larger fraction of $\tilde{\mathcal{P}}_n$ is required to ensure a relative error bound of $\Delta_{\mathcal{E}} \leqslant 0.01$; however, in every case this can still be achieved by only considering between 1–10% of $\tilde{\mathcal{P}}_n$. Interestingly, for higher p, increasing η reduces the number of errors that need to be considered, while for lower p, this trend is reversed. Figure 2

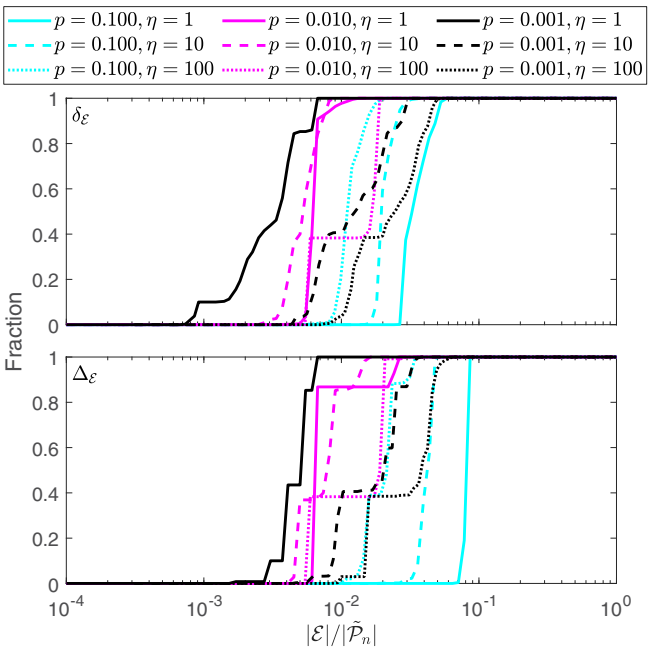

FIG. 1. The fraction of 1 000 randomly generated [[7,1]] codes that yield a relative error $\delta_{\mathcal{E}} \leqslant 0.01$ or relative error bound $\Delta_{\mathcal{E}} \leqslant 0.01$ for varying $|\mathcal{E}|$ and biased XZ channel parameters.

shows the results of a similar analysis for codes with $5 \leqslant n \leqslant 7$ and $1 \leqslant k \leqslant 3$ on a biased XZ channel with $p = 0.01$ and $\eta = 10$. It can be seen that increasing k for fixed n reduces the fraction of errors that must be considered, which makes sense given that encoding a larger number of qubits will lead

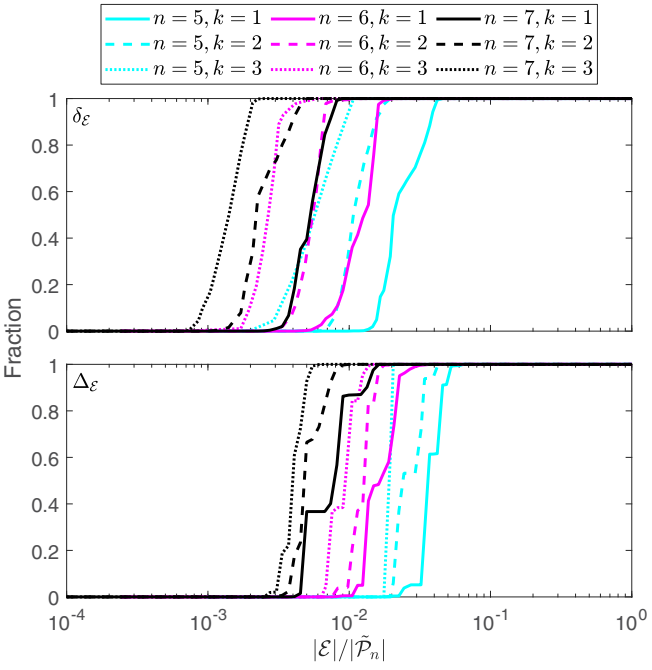

FIG. 2. The fraction of 1 000 randomly generated [[$5 \leqslant n \leqslant 7$, $1 \leqslant k \leqslant 3$]] codes that yield a relative error $\delta_{\mathcal{E}} \leqslant 0.01$ or relative error bound $\Delta_{\mathcal{E}} \leqslant 0.01$ for a biased XZ channel ($p = 0.01$ and $\eta = 10$) and varying $|\mathcal{E}|$.

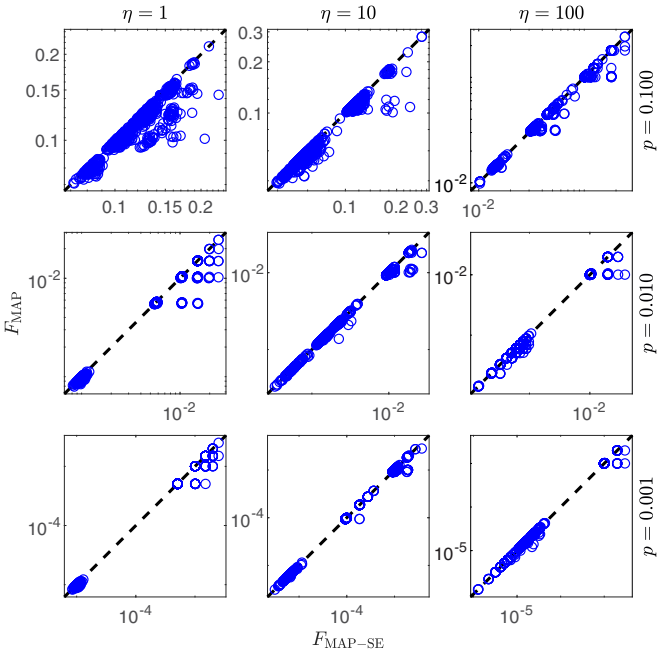

FIG. 3. F_{MAP} versus $F_{\mathrm{MAP-SE}}$ for 1 000 random [[7,1]] codes on biased XZ channels with varying parameters. The dotted lines give $F_{\mathrm{MAP}} = F_{\mathrm{MAP-SE}}$.

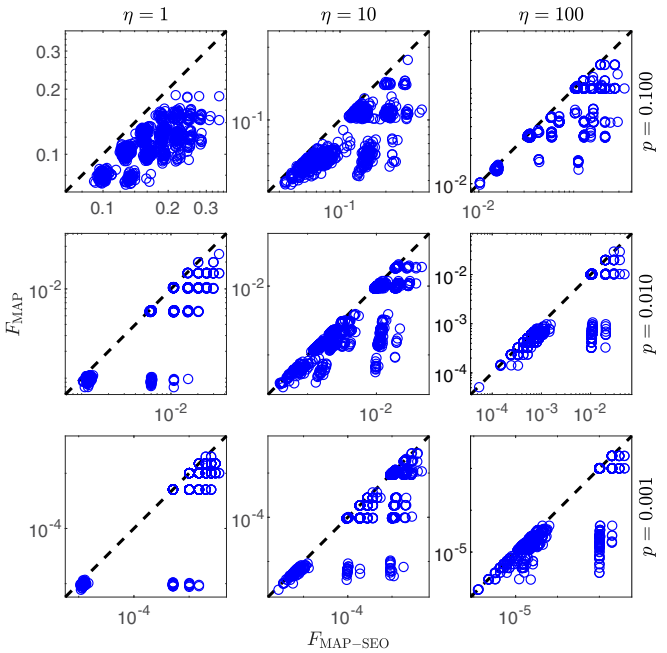

FIG. 4. F_{MAP} versus $F_{\mathrm{MAP-SEO}}$ for 1 000 random [[7,1]] codes on biased XZ channels with varying parameters. The dotted lines give $F_{\mathrm{MAP}} = F_{\mathrm{MAP-SEO}}$.

to a higher FER. Furthermore, increasing n for fixed k reduces the fraction of errors that need to be considered, which bodes well for the analysis of longer codes. We note that changing p and/or η has little effect on these observations.

B. Most likely error

We now consider the decoder of Eq. (34) that determines the single most likely error given a syndrome measurement, which has an error rate as given in Eq. (35). Note that $F_{\mathrm{MAP-SE}}$ is simpler to calculate than F_{MAP} as it does not require a complete partitioning of $\tilde{\mathcal{P}}_n$ to form $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$. When using $F_{\mathrm{MAP-SE}}$ as an approximation of F_{MAP}, the best case scenario is that the most likely coset $\hat{A}_z$ will contain $\tilde{\hat{E}}_z$ for every z, which gives $F_{\mathrm{MAP-SE}} = F_{\mathrm{MAP}}$. In the worst case scenario, two things will occur. Firstly, the probability distributions over every $\hat{A}_z$ will be uniform; that is, $P(\hat{A}_z)/|\tilde{\mathcal{S}}| = P(\hat{A}_z)/2^{n-k}$ for all z. Secondly, the distributions over every $\tilde{\hat{E}}_z\tilde{\mathcal{S}}$ will be sharply peaked without $P(\tilde{\hat{E}}_z\tilde{\mathcal{S}})$ being large; that is, for every z, $P(\tilde{\hat{E}}_z) = P(\hat{A}_z)/2^{n-k} + \varepsilon$ and $P(\tilde{\hat{E}}_z\tilde{\mathcal{S}}\backslash\tilde{\hat{E}}_z) = \varepsilon'$ for some small $\varepsilon, \varepsilon' \geqslant 0$. In general, it is therefore the case that

$$F_{\mathrm{MAP}} \leqslant F_{\mathrm{MAP-SE}} < 1 - \frac{1 - F_{\mathrm{MAP}}}{2^{n-k}}. \tag{51}$$

This upper bound on $F_{\mathrm{MAP-SE}}$ is very loose, and in practice, $F_{\mathrm{MAP-SE}}$ tends to be quite close to F_{MAP}. To demonstrate this, we have again constructed 1 000 random [[7,1]] codes. For each code, we have then determined both F_{MAP} and $F_{\mathrm{MAP-SE}}$ for the same nine biased XZ channel parameter combinations considered in Sec. III A ($p = 0.1$, 0.01, or 0.001 and $\eta = 1$, 10, or 100). The results of this are shown in Fig. 3. Especially for the codes yielding a low F_{MAP}, which are the codes of greatest interest, it can be seen that the difference between $F_{\mathrm{MAP-SE}}$ and F_{MAP} is often negligible.

$F_{\mathrm{MAP-SE}}$ can itself be approximated using a limited error set $\mathcal{E}$. We call this approximation $F_{\mathcal{E}-\mathrm{SE}}$, and it can be calculated in much the same manner as $F_{\mathcal{E}}$. Again, $\mathcal{E}$ is first partitioned by syndrome to give $B_1, \ldots, B_r$. For each $1 \leqslant i \leqslant r$, we then determine the most likely error $\tilde{\hat{E}}_i \in B_i$, which we use to define $\hat{A}_i = \{\tilde{E} \in B_i : \tilde{\hat{E}}_i\tilde{E} \in \tilde{\mathcal{S}}\}$. With this altered definition of $\hat{A}_i$, $F_{\mathcal{E}-\mathrm{SE}}$ is given by the right-hand side of Eq. (44). Furthermore, the relative error bound of Eq. (47) also holds for $F_{\mathcal{E}-\mathrm{SE}}$ with respect to $F_{\mathrm{MAP-SE}}$. We emphasize that $F_{\mathcal{E}-\mathrm{SE}}$ can be calculated faster than $F_{\mathcal{E}}$ as there is no need to fully partition each B_i.

C. Most likely error only

As outlined in Sec. II D, the single most likely error decoder for an $[[n, k]]$ stabilizer code can be viewed as a decoder for an associated $[2n, n+k]$ classical code $\mathcal{C}$. However, the calculation of $F_{\mathrm{MAP-SE}}$ as in Eq. (35) is more complicated than determining the FER of a classical MAP decoder as the cosets $\tilde{\hat{E}}_z\tilde{\mathcal{S}}$ still need to be enumerated. If we ignore the coset nature of the error correction, then we get

$$F_{\mathrm{MAP-SEO}} = 1 - \sum_{z \in \mathrm{GF}(2)^{n-k}} P(\tilde{\hat{E}}_z), \tag{52}$$

where "SEO" stands for "single error only." Note that this is exactly the FER of the classical decoder for $\mathcal{C}$ as in Eq. (10). Given the nature of the assumptions leading to Eq. (51), it also holds for $F_{\mathrm{MAP-SEO}}$. Again, it is a very loose upper bound, and as can be seen in Fig. 4, $F_{\mathrm{MAP-SEO}}$ does tend to be somewhat close to F_{MAP}. In particular, it can be seen that the

codes yielding a minimal value of $F_{\mathrm{MAP-SEO}}$ also often yield a near-minimal value of F_{MAP}.

$F_{\mathrm{MAP-SEO}}$ can also be approximated using a limited error set to yield $F_{\mathcal{E}-\mathrm{SEO}}$. This involves first partitioning $\mathcal{E}$ to form $B_1, \ldots, B_r$ and then determining the most likely error $\tilde{\hat{E}}_i$ in B_i. By defining $\hat{A}_i = \tilde{\hat{E}}_i$, $F_{\mathcal{E}-\mathrm{SEO}}$ is also given by the right-hand side of Eq. (44). Note that as no partitioning of each B_i is required, calculating $F_{\mathcal{E}-\mathrm{SEO}}$ is less complex than calculating $F_{\mathcal{E}-\mathrm{SE}}$ (or, indeed, $F_{\mathcal{E}}$). The upper bound on relative error given in Eq. (47) again holds for $F_{\mathcal{E}-\mathrm{SEO}}$ with respect to $F_{\mathrm{MAP-SEO}}$. Assuming that $\mathcal{E}$ contains the most likely errors in $\tilde{\mathcal{P}}_n$, which is the case for the construction given in Sec. III A, we can derive another simple bound. In particular, if $\mathcal{E}$ contains errors corresponding to r different syndromes, then an error $\tilde{E}' \notin \mathcal{E}$ yielding one of the other $2^{n-k} - r$ possible syndromes must have probability $P(\tilde{E}') \leqslant \min_{\tilde{E}\in\mathcal{E}} P(\tilde{E})$ (as otherwise it would be an element of $\mathcal{E}$). This gives

$$F_{\mathcal{E}-\mathrm{SEO}} - F_{\mathrm{MAP-SEO}} \leqslant (2^{n-k} - r) \min_{\tilde{E}\in\mathcal{E}} P(\tilde{E}) = \alpha, \tag{53}$$

which leads to a combined bound on the relative error of

$$\begin{aligned} \delta_{\mathcal{E}-\mathrm{SEO}} &= \frac{F_{\mathcal{E}-\mathrm{SEO}} - F_{\mathrm{MAP-SEO}}}{F_{\mathrm{MAP-SEO}}} \\ &\leqslant \frac{\min[1 - P(\mathcal{E}), \alpha]}{F_{\mathcal{E}-\mathrm{SEO}} - \min[1 - P(\mathcal{E}), \alpha]} \qquad (54) \\ &= \Delta_{\mathcal{E}-\mathrm{SEO}}. \qquad (55) \end{aligned}$$

IV. CODE PERFORMANCE

In this section, we employ the approximate FER calculation methods outlined in Sec. III to investigate the performance of various families of codes on biased XZ and AD channels. There is a particular focus on the performance of cyclic codes as it has previously been shown that a [[7,1,3]] cyclic code with $\mathcal{S} = \langle XZIZXII\rangle_{\mathrm{cyc}}$ performs near optimally on the biased XZ channel for a range of error probabilities and biases [4].

A. [[7,1]] codes

To demonstrate our approach, we first consider the case of [[7,1]] codes. We have constructed all of the [[7,1]] cyclic codes by enumerating the self-orthogonal additive cyclic $(7, 2^6)_4$ codes as outlined in Sec. II B. There are 11 such codes, six of which are inequivalent. Following the lead of Ref. [4], we have also constructed 10 000 random codes to serve as a point of comparison. Our random construction, as detailed in Sec. III A, differs to that of Ref. [4] in that we do not require our codes to have weight-four generators or distance $d \geqslant 3$. For both biased XZ and AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000, we have determined $F_{\mathcal{E}}$ for each code, ensuring that in every case, $\mathcal{E}$ is large enough to give $\Delta_{\mathcal{E}} \leqslant 0.01$. This can be achieved without having to construct a new $\mathcal{E}$ for every FER calculation. For some channel type (biased XZ or AD), channel parameter combination (p and η pair), and code family (random or cyclic), we first construct $\mathcal{E}$, as outlined in Sec. III A, such that $1 - P(\mathcal{E}) \leqslant 0.1$ and then calculate $F_{\mathcal{E}}$ for every code in the family. If $\Delta_{\mathcal{E}} > 0.01$ for any of these codes, we then add errors to $\mathcal{E}$ until $1 - P(\mathcal{E}) \leqslant 0.01$ and recalculate $F_{\mathcal{E}}$ for these codes. This proceeds iteratively, reducing $1 - P(\mathcal{E})$ by a factor of 10 each time, until $\Delta_{\mathcal{E}} \leqslant 0.01$ for every code.

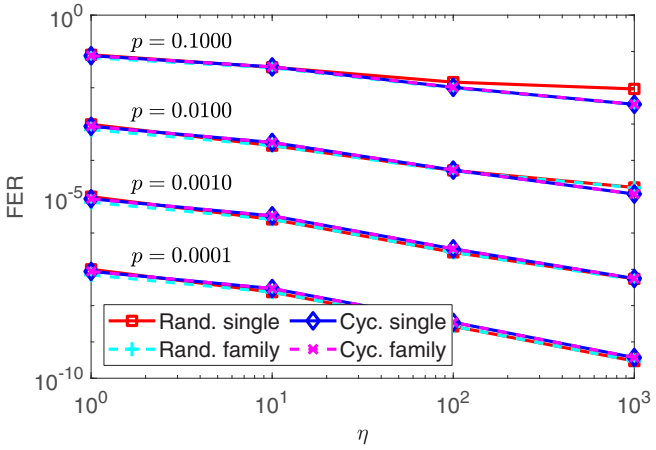

FIG. 5. FER performance of the best cyclic and random [[7,1]] codes on biased XZ channels.

For each channel type, channel parameter combination, and code family, we report two values. The first of these is simply the lowest FER of any code in the family, which can be viewed as a performance measure of the family as a whole. The second is the FER of the code that performs the best on average across all channel parameter combinations. We quantify this average performance by taking the geometric mean of a code's FERs across the associated channels. That is, we take the best code to be the one with stabilizer

$$\mathcal{S}_{\mathrm{best}} = \operatorname*{argmin}_{\mathcal{S}\in\mathcal{F}} \left(\prod_{i=1}^{N} F_{\mathcal{E}_i}^{\mathcal{S}} \right)^{1/N}, \tag{56}$$

where $\mathcal{F}$ is the family of stabilizers and $\mathcal{E}_i$ is the error set associated with one of the $N = 16$ channels. Figure 5 shows these values for the biased XZ channel. It can be seen that for every parameter combination, there is a cyclic code that performs nearly as well as the best random code. Furthermore, there is a single cyclic code that performs optimally (among the cyclic codes) on all channels. In fact, there are three such codes; however, they are all equivalent to the code with stabilizer $\mathcal{S} = \langle XZIZXII\rangle_{\mathrm{cyc}}$. The values for the AD channel are shown in Fig. 6, where the code with stabilizer $\langle XZIZXII\rangle_{\mathrm{cyc}}$ again performs optimally among the cyclic codes; however, in some cases, it is outperformed by the best random code by quite a margin, particularly at lower error probabilities (note that for consistency, we have used the same random codes for both channel types). At these low error probabilities, it can also be seen that unlike the biased XZ channel, increasing the bias does little to decrease the error rate. Interestingly, the code with stabilizer $\langle YZIZYII\rangle_{\mathrm{cyc}}$, which is not equivalent to $\langle XZIZXII\rangle_{\mathrm{cyc}}$, yields the same performance. This is a result of the fact that $p_X = p_Y$ for the AD channel, which means that applying the permutation $X \leftrightarrow Y$ to a code's stabilizer on any subset of qubits has no effect on its performance.

Note that the relative error of a geometric mean of FERs, such as the one in Eq. (56), is bounded by the relative error of

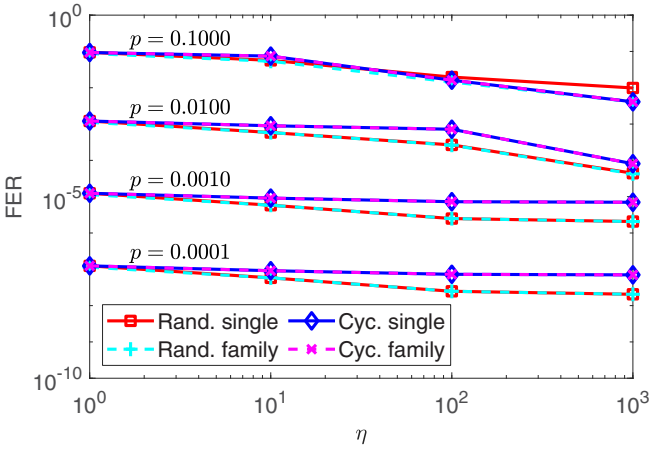

FIG. 6. FER performance of the best cyclic and random [[7,1]] codes on AD channels.

the least accurate individual FER. This follows from

$$\left(\prod_{i=1}^{N} F_{\mathcal{E}_i}\right)^{1/N} \leqslant \left[\prod_{i=1}^{N}\left(1+\Delta_{\mathcal{E}_i}\right)F_{\mathrm{MAP}_i}\right]^{1/N} \leqslant \max_i\left(1+\Delta_{\mathcal{E}_i}\right)\left(\prod_{i=1}^{N} F_{\mathrm{MAP}_i}\right)^{1/N}, \tag{57}$$

which gives

$$\frac{\left(\prod_{i=1}^{N} F_{\mathcal{E}_i}\right)^{1/N} - \left(\prod_{i=1}^{N} F_{\mathrm{MAP}_i}\right)^{1/N}}{\left(\prod_{i=1}^{N} F_{\mathrm{MAP}_i}\right)^{1/N}} \leqslant \max_i \Delta_{\mathcal{E}_i}. \tag{58}$$

B. Other parameters

We have repeated the analysis of Sec. IV A for codes with $5 \leqslant n \leqslant 12$ and $1 \leqslant k \leqslant 3$. For each combination of n and k, this has again begun by constructing 10 000 random codes and enumerating the cyclic stabilizer codes. The number of these cyclic codes is given in the first column of Table I. The first value in each row gives the number of inequivalent codes, while the value in brackets gives the total number of distinct codes. Note that for odd n, the number of distinct codes we report is consistent with Ref. [17]. To the best of our knowledge, neither the number of distinct codes with even n or the number of inequivalent codes with any n has previously been published (Ref. [18] does give total number of distinct $[[n, k \leqslant n]]$ cyclic codes, but it does not include the number for each specific k). Note that in some cases, there are no cyclic codes.

For each channel type, code family, and pair of n and k, we report two values. The first of these is the geometric mean of the FERs for the single best code as defined in Eq. (56); that is,

$$\lambda = \min_{\mathcal{S}\in\mathcal{F}}\left(\prod_{i=1}^{N} F_{\mathcal{E}_i}^{\mathcal{S}}\right)^{1/N}. \tag{59}$$

TABLE I. The number of inequivalent (distinct) $[[n, k]]$ cyclic codes, single-generator cyclic codes, cyclic codes with weight-four generators, cyclic CSS codes, dual-containing CSS codes, and linear cyclic codes.

$[[n, k]]$	Cyc.	One gen.	$w = 4$	CSS	DC CSS	Lin.
[[5,1]]	4 (5)	4 (5)	4 (5)	2 (2)	0	1 (2)
[[5,2]]	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
[[5,3]]	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
[[6,1]]	21 (21)	18 (18)	15 (15)	6 (6)	0	0 (0)
[[6,2]]	35 (42)	30 (36)	17 (21)	9 (9)	2	2 (3)
[[6,3]]	12 (15)	12 (15)	3 (6)	4 (4)	0	0 (0)
[[7,1]]	6 (11)	5 (9)	6 (11)	3 (4)	1	1 (2)
[[7,2]]	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
[[7,3]]	15 (54)	15 (54)	0 (0)	4 (8)	0	0 (0)
[[8,1]]	57 (87)	30 (48)	24 (33)	8 (8)	0	0 (0)
[[8,2]]	46 (79)	27 (48)	19 (25)	7 (7)	3	1 (1)
[[8,3]]	33 (63)	21 (48)	12 (15)	6 (6)	0	0 (0)
[[9,1]]	15 (27)	15 (27)	9 (21)	4 (4)	1	0 (0)
[[9,2]]	15 (27)	15 (27)	0 (0)	4 (4)	0	0 (0)
[[9,3]]	5 (9)	5 (9)	3 (3)	2 (2)	1	0 (0)
[[10,1]]	42 (63)	39 (60)	21 (33)	6 (6)	0	0 (0)
[[10,2]]	14 (21)	13 (20)	11 (15)	3 (3)	6	2 (3)
[[10,3]]	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
[[11,1]]	9 (33)	9 (33)	9 (33)	2 (2)	2	0 (0)
[[11,2]]	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
[[11,3]]	0 (0)	0 (0)	0 (0)	0 (0)	3	0 (0)
[[12,1]]	300 (465)	162 (288)	51 (75)	20 (20)	0	0 (0)
[[12,2]]	536 (768)	288 (432)	65 (81)	35 (35)	11	2 (3)
[[12,3]]	312 (528)	198 (360)	27 (30)	26 (26)	0	0 (0)

The second value is the geometric mean of the minimum FERs of all codes in a family for each channel; that is,

$$\mu = \left(\prod_{i=1}^{N} \min_{\mathcal{S}\in\mathcal{F}} F_{\mathcal{E}_i}^{\mathcal{S}}\right)^{1/N}, \tag{60}$$

which can again be viewed as a performance measure of the family as a whole. Figure 7 shows these values for the biased XZ channel. It can be seen that for both the random and cyclic codes, there is typically a single code that performs nearly as well as the family as a whole across the 16 different channels considered. Furthermore, when $[[n, k]]$ cyclic codes exist, there is often one that performs as well as or better than the best random code we have created. In fact, for $n \geqslant 9$ and $k = 1$, the best cyclic codes significantly outperform the best random codes. The results for the AD channel are given in Fig. 8. Again, where $[[n, k]]$ cyclic codes exist, they typically perform favorably compared to the random codes. However, any performance advantages over the random codes are less pronounced than in the biased XZ case.

Generators for the best cyclic codes on both the biased XZ and AD channels can be found in Table II (for reference, we also give their distances). In particular, we list generators for all codes that yield a geometric mean of FERs within 1% of the minimum value we have observed (these are all codes that could conceivably be optimal within our margin of error). There are a few notable properties of these codes. The first of these is that they can all be expressed using a single generator. While, as shown in the second column of Table I, a

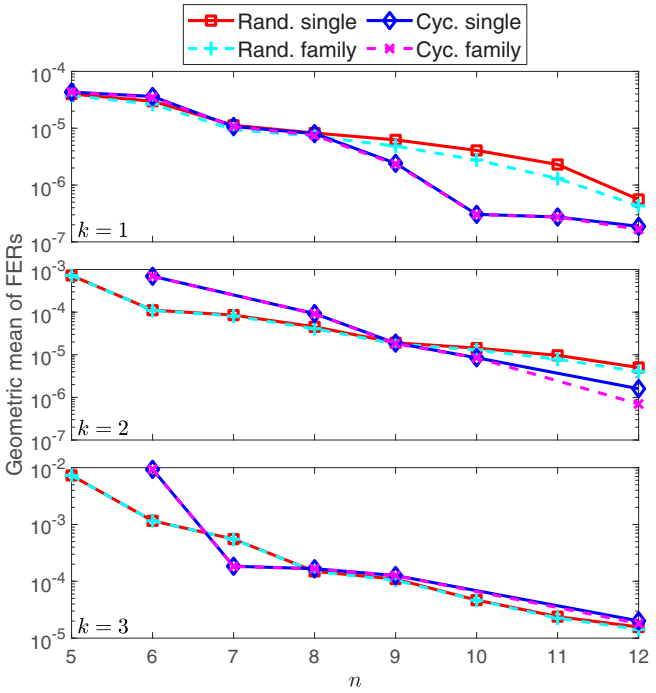

FIG. 7. The geometric mean of FERs for codes on biased XZ channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000.

large number of codes have such a representation, this is still a somewhat surprising result. It can also be seen that in nearly every case, there are codes that perform well for both the biased XZ and AD channels (the only exceptions to this are the [[6,1]], [[6,2]], and [[10,2]] cases). A third property of note is that the codes for the AD channel typically come in pairs,

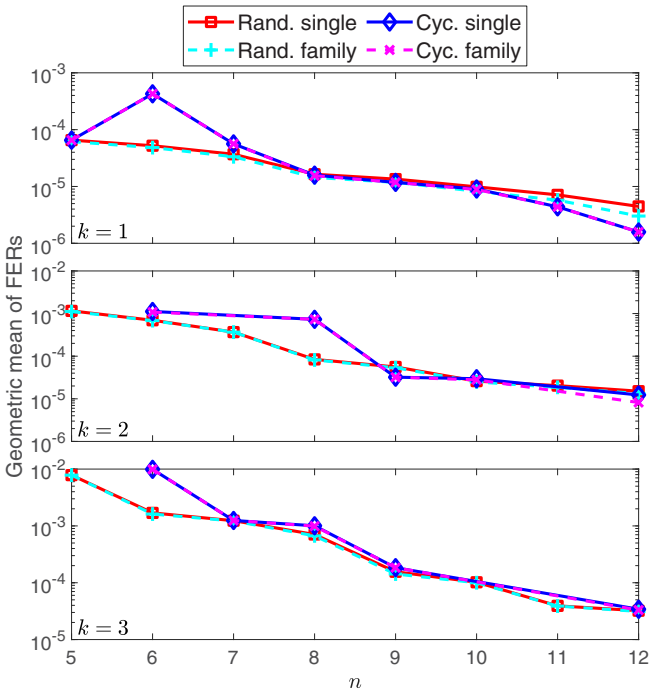

FIG. 8. The geometric mean of FERs for codes on AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000.

TABLE II. Generators and distances for the best performing inequivalent cyclic codes on the biased XZ and AD channels. Note that each stabilizer can be expressed using a single generator; that is, each generator given corresponds to a different code. The generators of codes performing well on both channel types are given in bold.

$[[n, k]]$	Biased XZ	AD
[[5,1]]	***YZIZY***, $d = 3$	***YZIZY***, $d = 3$
[[6,1]]	*YIZZIY*, $d = 2$	*XZZZZX*, $d = 2$ *YZZZZY*, $d = 2$
[[6,2]]	*YZIZYI*, $d = 2$	*XIZIXY*, $d = 2$ *YIZIYX*, $d = 2$ *XZIZXY*, $d = 2$ *YZIZYX*, $d = 2$
[[6,3]]	***XZXXZX***, $d = 2$ ***XZZXZZ***, $d = 2$ *XIYXIY*, $d = 2$ *YZIYZI*, $d = 2$	***XZXXZX***, $d = 2$ ***XZZXZZ***, $d = 2$ *YZYYZY*, $d = 2$ *YZZYZZ*, $d = 2$
[[7,1]]	***XZIZXII***, $d = 3$	***XZIZXII***, $d = 3$ *YZIZYII*, $d = 3$
[[7,3]]	***XZZZXZX***, $d = 2$ *XZIIYZY*, $d = 2$ *YIIZYZX*, $d = 2$	***XZZZXZX***, $d = 2$ *YZZZYZY*, $d = 2$
[[8,1]]	***YIIZIZZX***, $d = 3$ *ZZYIIIIY*, $d = 3$	***YIIZIZZX***, $d = 3$ *XIIZIZZY*, $d = 3$
[[8,2]]	***YIIXIIYX***, $d = 2$ ***YIZZIIXZ***, $d = 2$ ***XIIYZIYY***, $d = 2$ *YIIZIIYZ*, $d = 2$ *YZIZIZYZ*, $d = 2$ *XZZZZZXZ*, $d = 2$	***YIIXIIYX***, $d = 2$ ***YIZZIIXZ***, $d = 2$ ***XIIYZIYY***, $d = 2$ *XIIYIIXY*, $d = 2$ *YIIXZIXX*, $d = 2$ *XIZZIIYZ*, $d = 2$
[[8,3]]	***YIXIIYZY***, $d = 2$ ***XZIIZXYY***, $d = 2$ *YZIZIXYX*, $d = 2$	***YIXIIYZY***, $d = 2$ ***XZIIZXYY***, $d = 2$ *XIYIIXZX*, $d = 2$ *YZIIZYXX*, $d = 2$
[[9,1]]	***ZIZYIIIIY***, $d = 3$	***ZIZYIIIIY***, $d = 3$ *ZIZXIIIIX*, $d = 3$
[[9,2]]	***IZIXIZIYY***, $d = 3$	***IZIXIZIYY***, $d = 3$ *IZIYIZIXX*, $d = 3$
[[9,3]]	***YZZIZZYII***, $d = 3$	***YZZIZZYII***, $d = 3$ *XZZIZZXII*, $d = 3$
[[10,1]]	***YZIZIIZIZY***, $d = 4$	***YZIZIIZIZY***, $d = 4$ *XZIZIIZIZX*, $d = 4$
[[10,2]]	*YZZIIIZZYI*, $d = 2$	*IYXIIIIIXY*, $d = 3$
[[11,1]]	***IYIIZIIZIIY***, $d = 3$	***IYIIZIIZIIY***, $d = 3$ *IXIIZIIZIIX*, $d = 3$
[[12,1]]	***YIXIXIIIIIZX***, $d = 4$	***YIXIXIIIIIZX***, $d = 4$ *XIYIYIIIIIZY*, $d = 4$
[[12,2]]	***IIZZIIXZZIXY***, $d = 4$ *YXZIXIIIIIYX*, $d = 4$	***IIZZIIXZZIXY***, $d = 4$ *IIZZIIYZZIYX*, $d = 4$
[[12,3]]	***ZZXIYIIIIYIX***, $d = 3$ *IZZIXIZZIYXY*, $d = 3$	***ZZXIYIIIIYIX***, $d = 3$ *ZZYIXIIIIXIY*, $d = 3$

one being an $X \leftrightarrow Y$ permuted version of the other. This is to be expected given the partial channel symmetry outlined in Sec. IV A. The only two exceptions to this are the [[5,1]] and [[10,2]] cases, where the single code given is invariant under an $X \leftrightarrow Y$ permutation (up to a permutation of qubit labels).

C. Hill climbing

The results of Sec. IV B, particularly those for $[[n \geqslant 9, 1]]$ codes on the biased XZ channel, show that constructing 10 000 random codes is not a reliable way of finding a good code for larger n. One approach to find better codes would be to simply increase the size of the random search. However, even with the reduction in error set size afforded by the approach of Sec. III A, this quickly becomes computationally impractical. As such, we need a more efficient search strategy. To achieve this, we use the observation of Sec. III C that codes yielding a low $F_{\mathcal{E}-\mathrm{SEO}}$ tend to also yield a low $F_{\mathcal{E}}$ (recall that $F_{\mathcal{E}} \leqslant F_{\mathcal{E}-\mathrm{SEO}}$). We can therefore reduce the search to finding codes that yield a low $F_{\mathcal{E}-\mathrm{SEO}}$, which is beneficial as it is typically several orders of magnitude faster to calculate $F_{\mathcal{E}-\mathrm{SEO}}$ than it is to calculate $F_{\mathcal{E}}$ to the same accuracy.

We start by considering the problem of finding codes that perform well for a single channel parameter combination. That is, we want to find a stabilizer $\mathcal{S}$ that yields a low $F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$. We have found a simple hill-climbing search strategy to be effective at this. This involves first constructing $\mathcal{S}$ at random. $\mathcal{S}$ is then mutated (modified) somehow to produce $\mathcal{S}'$, and if $F^{\mathcal{S}'}_{\mathcal{E}-\mathrm{SEO}} \leqslant F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$, then $\mathcal{S}$ is replaced with $\mathcal{S}'$. This process repeats for a predetermined number of iterations, after which we calculate $F^{\mathcal{S}}_{\mathcal{E}}$ to quantify the actual performance of the code. Similar to the random search outlined in Sec. IV A, we ensure that the relative error of all approximate FER calculations is less than 1%. To achieve this, we again initially construct $\mathcal{E}$ such that $1 - P(\mathcal{E}) \leqslant 0.1$, and if $\Delta_{\mathcal{E}-\mathrm{SEO}} > 0.01$ ($\Delta_{\mathcal{E}} > 0.01$) for any calculation of $F_{\mathcal{E}-\mathrm{SEO}}$ ($F_{\mathcal{E}}$), then we add errors to $\mathcal{E}$ to reduce $1 - P(\mathcal{E})$ by a factor of 10 and recalculate the error rate. To better explore the space of possible stabilizers, we run a number of these hill-climbing instances in parallel (this is often called hill climbing with random restarts [38]).

The choice of a mutation operator that maps $\mathcal{S}$ to $\mathcal{S}'$ is limited by the requirement that $\mathcal{S}'$ must be a stabilizer. We consider two types of mutation that satisfy this constraint. The first of these involves permuting the nonidentity Pauli matrices of all stabilizer elements at any given index $1 \leqslant i \leqslant n$ with probability $1/n$. Note that these permutations correspond to a multiplication of coordinates of the associated classical GF(4) code by a nonzero scalar $\alpha \in \mathrm{GF}(4)$ followed by a possible conjugation. The second mutation method involves first removing any given generator M_i of $\mathcal{S} = \langle M_1, \ldots, M_{n-k} \rangle$ with probability $1/(n-k)$ and then adding generators, as outlined in Sec. III A, to form $\mathcal{S}'$. When performing this generator mutation, we still require that all qubits are involved in the stabilizer; if this is not achieved after adding the new generators, we remove them and try again. To compare these two mutation operators, we consider [[9,1]] codes on the biased XZ channel with $p = 0.01$ and $\eta = 10$. We have run 1 000 hill-climbing instances, each for a maximum of 1 000 iterations. Across all of these instances, Fig. 9 shows the 95th percentile $F_{\mathcal{E}-\mathrm{SEO}}$ at each iteration; that is, it shows the 50th lowest $F_{\mathcal{E}-\mathrm{SEO}}$ (we have chosen to show this value as it reflects the performance of the best codes while having less potential variance than showing the FER of the single best code). As a control, we have also tested random mutation, which involves simply creating $\mathcal{S}'$ at random (this reduces hill climbing to a random search). It can be seen that both the permutation and generator mutation outperform this random mutation, with the permutation mutation performing best initially but then tapering off somewhat. Finally, we have tested a combination of the two mutation methods (a generator mutation followed by a permutation mutation), which can be seen to perform better than either of the methods individually.

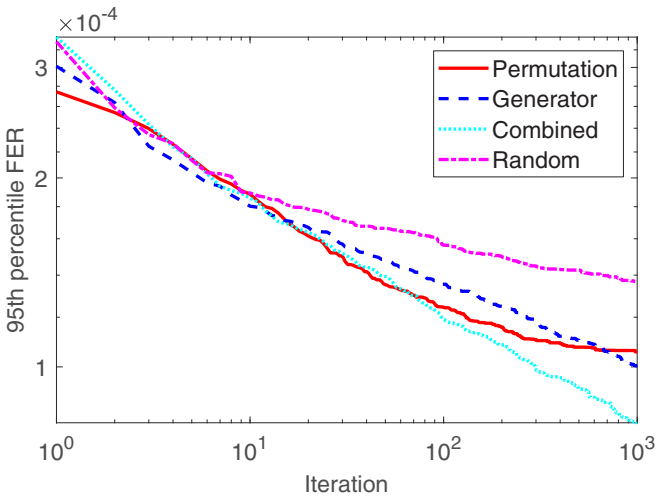

FIG. 9. The 95th percentile $F_{\mathcal{E}-\mathrm{SEO}}$ found by 1 000 hill-climbing instances based on various mutation methods for [[9,1]] codes on a biased XZ channel ($p = 0.01$ and $\eta = 10$).

D. Multiobjective hill climbing

The results of Sec. IV B suggest that there are typically codes that perform well across a range of channel parameter combinations. We can search for such codes by building on the hill-climbing algorithm outlined in Sec. IV C. In particular, instead of comparing $F^{\mathcal{S}'}_{\mathcal{E}-\mathrm{SEO}}$ to $F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$, we compute and compare the geometric means $(\prod_{i=1}^{N} F^{\mathcal{S}'}_{\mathcal{E}_i-\mathrm{SEO}})^{1/N}$ and $(\prod_{i=1}^{N} F^{\mathcal{S}}_{\mathcal{E}_i-\mathrm{SEO}})^{1/N}$ of the FERs for N channel parameter combinations. Following Eq. (58), we ensure that these geometric means are accurate to within 1% by keeping each of the individual $\Delta_{\mathcal{E}_i-\mathrm{SEO}} \leqslant 0.01$ as outlined in Sec. IV C. Again, we run a number of these hill-climbing instances in parallel, and at the end of each one, we calculate $(\prod_{i=1}^{N} F^{\mathcal{S}}_{\mathcal{E}_i})^{1/N}$. Note that for $N = 1$, this search reduces to that of Sec. IV C.

We have performed such searches for the same cases considered in Sec. IV B (that is, codes with $5 \leqslant n \leqslant 12$ and $1 \leqslant k \leqslant 3$ for biased XZ and AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000). For each combination of n, k, and channel type, we have run 1 000 hill-climbing instances based on the combined generator and permutation mutation, each for 1 000 iterations. Figure 10 compares the performance (that is, the geometric mean of FERs) of the best codes found in this way to that of the best cyclic codes (the other values shown will be detailed in Secs. IV E to IV G). It can be seen that in all but the [[10,1]] case, the best code found via hill climbing is either as good as or better than the best cyclic code. Very similar results can be seen in Fig. 11 for the AD channel, where the best code found via hill climbing performs as well as or better than the best cyclic code in every instance. Generators for the best codes

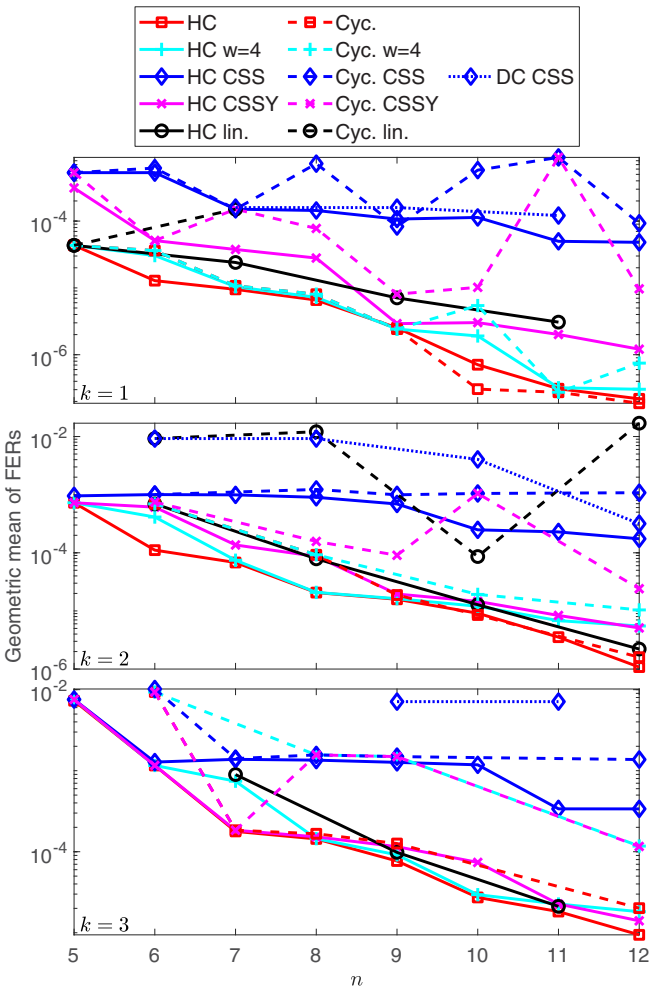

FIG. 10. The performance (geometric mean of FERs) of the best $[[5 \leqslant n \leqslant 12, 1 \leqslant k \leqslant 3]]$ codes found via hill climbing for biased XZ channels with with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000. Also shown is the performance of the best cyclic codes and dual-containing CSS codes.

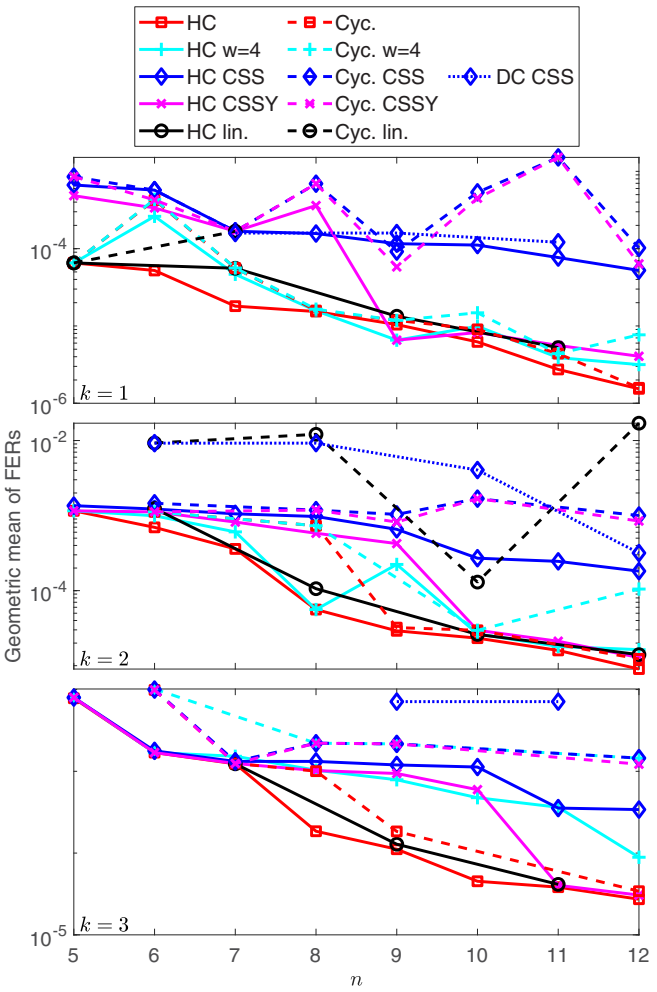

FIG. 11. The performance (geometric mean of FERs) of the best $[[5 \leqslant n \leqslant 12, 1 \leqslant k \leqslant 3]]$ codes found via hill climbing for AD channels with with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000. Also shown is the performance of the best cyclic codes and dual-containing CSS codes.

we have found for the biased XZ and AD channels can be found in Tables III and IV, respectively.

E. Weight-four codes

Through slight modification of the hill-climbing algorithm, we can search for good codes that satisfy structure constraints. The first constraint we consider is the requirement that the stabilizer has a representation involving only weight-four generators; such codes are of practical interest as their syndrome measurements involve fewer qubits, and are hence less complex, than those for codes with high-weight generators. The first modification required to search for these codes, which is somewhat obvious, is to ensure the initial random stabilizer has weight-four generators. This also extends to the generator permutation; that is, any generator added to replace a removed one must also have weight four. No change to the permutation mutation is required as it preserves the weight of stabilizer elements. We compare the codes found via this constrained hill-climbing search to the cyclic codes with a weight-four generator representation. The number of such cyclic codes is given in the third column of Table I, where it can be seen that they are reasonably plentiful.

The performance of the weight-four codes found via hill climbing for the biased XZ channel is shown in Fig. 10. It can be seen that in a lot of cases, these codes perform nearly as well as those found using unconstrained hill climbing in Sec. IV D. The performance of the weight-four cyclic codes is more varied. In some cases, they are optimal (among the cyclic codes), while in others, they perform relatively poorly. Figure 11 shows that the performance of the weight-four codes found via hill climbing for the AD channel is somewhat mixed, ranging from outperforming the unconstrained [[9,1]] codes to performing very poorly for $k = 3$ and $n \geqslant 8$. The performance of the weight-four cyclic codes relative to the best unconstrained cyclic codes is much the same as for the biased XZ channel. Generators for the best weight-four codes found via hill climbing can be found in Tables V and VI, and generators for the best cyclic codes are given in Table VII.

TABLE III. Generators and distances for the best codes found for the biased XZ channel using hill climbing.

$n\backslash k$	1	2	3
5	*IXXZZ* *YZYIZ* *IZZYY* *XZIZX* $d = 3$	*XXYZI* *XXZXZ* *XIIZY* $d = 1$	*XYZYZ* *IIIXX* $d = 1$
6	*IXXXYZ* *YIYIZZ* *IYYIII* *XZXIYX* *ZZXYIZ* $d = 3$	*ZIZYXY* *XXZXYX* *IYYZYZ* *XZIZYX* $d = 2$	*IIYYZY* *XXZYZZ* *XIIIYX* $d = 1$
7	*XZZXYYI* *XYYXZXY* *XIXZZZZ* *YZIZXXY* *ZYIXYIY* *YIYZYZY* $d = 3$	*IIZYXZY* *ZIXYIXY* *XIZZYXZ* *XYIXYIY* *ZXZXYII* $d = 2$	*YZZYIXX* *ZYXZIZY* *XYIXXIY* *ZZXXZXZ* $d = 2$
8	*ZIIYXIYX* *ZYYIYYXX* *IZYIYXXI* *XYXZXZII* *YIYYZIZY* *ZXXXYIII* *YXXZYYZX* $d = 3$	*YYIIIXYI* *ZIXYZZIY* *ZYXZYXXX* *IZXYIZYI* *YZZZIZIX* *IXXIZIXI* $d = 2$	*XZZZYYIY* *YZIYXIXX* *XZXXYYYI* *IYZYIYZI* *YYZZZXIY* $d = 2$
9	*YXYXIIIII* *YZIYIZIZI* *XYIIYIIIX* *IXIXIZZZZ* *XYXYIXZXI* *XIXXXYYYY* *XXYYYYXXZ* *YZXXZYZIZ* $d = 3$	*XIZZZZXYX* *IIZIYYIIX* *IXYYZZYYZ* *IIZXXXXZI* *ZZXIXXIXY* *YXXZIZIYI* *ZYYYIXZXI* $d = 2$	*ZIZZXXYIZ* *ZIIYXIIYY* *IYZXXXIIX* *ZXZYYZZZI* *YZYYIYYXI* *YYZIZZXII* $d = 2$
10	*XYYXYIXYXX* *XYYYIZXZYY* *ZIXZXYIZYI* *YXYYXXIIIX* *IZZZIIZYYZ* *XXZXYZYXYZ* *XXIZXIXXIX* *ZXXIYYIXIX* *XZXZYYXXXI* $d = 3$	*ZXZIXZIIIY* *YIIXYZIZII* *XIYIIIZXZY* *ZIXIYYIIYX* *XZZYIXIYXX* *YYIXYXYXII* *ZZYZZXXZXI* *ZXZYXZYYZI* $d = 3$	*ZZXXIIXZZY* *IYZYZZYIZY* *YZIYXZZZIX* *ZXYZIIYXXY* *ZYXIXZIYZY* *XZYXYYIYXY* *YYYYIZXIII* $d = 2$
11	*IZXZXZXXIZX* *ZXIZXYIIIYY* *ZXYIIXYYYXI* *YIYZXXZIYXX* *IYZYXXIYZYX* *IYXXYXIYXZZ* *ZIXZYIZXZIX* *YYXZXYXYZXX* *ZIYZXYXZIYI* *YYZIXZZIZIZ* $d = 3$	*YZXIXIZYXZX* *ZXIIYIXYZXI* *YYXZXYIIXXX* *ZXIIZXZYYZX* *IXZYIIXYIIZ* *ZXIZIXYXIXY* *IZIXZZZYXXX* *ZYXIZYXXXXX* *XZIZIIIXYIY* $d = 3$	*YZXZXXZXXYY* *YZIYXIIYIXX* *YYIZYXYZXXX* *ZYIZIIYZYX* *YYZIIYXIZXY* *YXZXYXZZYII* *YIIIIZYZIYX* *IZXXYXZIYIX* $d = 2$
12	*YYIIXIYYIXZZ* *YXIYZIXZIZIY* *XXXXIZXIZXXZ* *ZXZYIIIIZYXI* *ZYYYZIZIYIZX* *IZYYXXXYIYIY* *ZYYYYIYYXIYX* *ZYIZYXIXIYXY* *IZIXIXYXXYYZ* *IXXZZYIIXXXY* *ZZIZYXIZXYZX* $d = 4$	*ZXZXZYXXZZYI* *ZIZYZIXIXZIY* *IYYYXZXXZYYY* *IXIIIYYZIZYZ* *IIYIIXYYZIIZ* *IIIXYYIZXYIX* *ZIXYIIZXIZZY* *YIYZZXYZIXXX* *YYXXYXXZYYYI* *YZYXZZYYYYII* $d = 3$	*IIYYYIZXIXII* *IZXZZXXIIXZZ* *ZZZYZXYIZZII* *YZIXIZXIIIXX* *XYZZZIXZYYYI* *XIYZYXIXXYIY* *YIZZYXXYZIII* *ZYXZXIIZIIZI* *ZXZZYYXZXIYY* $d = 3$

TABLE IV. Generators and distances for the best codes found for the AD channel using hill climbing.

$n\backslash k$	1	2	3
5	*YYXIX* *IXYYX* *IZXXZ* *XZIZX* $d = 3$	*YXZZX* *IIYYX* *IYIYX* $d = 1$	*XZZYX* *IXIXI* $d = 1$
6	*XZZZIY* *IZYXYY* *YIZXZI* *YYXIXZ* *IZZYXZ* $d = 3$	*XYIYZY* *XYYIXZ* *IIXXZZ* *XZYZZI* $d = 1$	*YIXIYZ* *IYXYYI* *IZXYZY* $d = 1$
7	*XZZIYIX* *XIYZYXZ* *IYYXXXZ* *ZIXXIYI* *XZZZIZI* *XIXYZIX* $d = 3$	*ZYYXYXX* *XYZZIZI* *XZIZZYZ* *YZZXIXY* *XYIYXXZ* $d = 2$	*IZXIZYX* *YZYXIIZ* *XYIIZIZ* *IYYXXIY* $d = 1$
8	*ZXXIYIYX* *YXXZXIIX* *YIYZIIXZ* *IXXXXYIZ* *ZYXZIZZX* *YIZXYZZY* *IYXZYZZI* $d = 3$	*YIYYYXIZ* *YYXXZYXZ* *ZYXIXYIZ* *ZZZXXIIZ* *YZIIXXXY* *IXIIXIZY* $d = 3$	*XZXIZYZI* *ZXXYXXZY* *YZZIXXIZ* *XIXXXZIX* *IZXYIZYZ* $d = 3$
9	*ZIXYXIYYI* *ZYYZXXYIX* *IYIXIYIZI* *YYZZIXIZI* *IXYYYXYYY* *IIIZIXZIY* *ZXZYIYYZX* *ZZXZZXIYY* $d = 3$	*IIYYXYIYI* *XZYXXZXII* *IXIZXXYIZ* *YZXXIYZYI* *YXZYXYZXY* *YZYYZIYZX* *YYYIYZYIX* $d = 3$	*IXIYZYZZI* *YZIIYZZXZ* *ZYIZIYIIX* *YZXXIXYII* *YXIIIIXYX* *YZZIXYIZX* $d = 3$
10	*YIIZYYXYYY* *XYZXIZZXYZ* *IXYXZYYYYX* *XYIIZZXXXZ* *XYIYIYXXIX* *XZZYZYYIXX* *ZIXIYZYZII* *IIXXXIZIXI* *XXYYXXXXZI* $d = 3$	*YZXYZYYIYY* *YYXZIIXZXI* *XIIYIIXXYI* *IYZZIIXXXX* *YXZIIXZIYY* *XXYYZIZZYI* *YYZYZZYIYZ* *YXYXXIZYYX* $d = 3$	*ZZZZYZZXZX* *XXZXZZXXYX* *XXYZXZXZXY* *YIZYIXZXYX* *ZXYYZXIYYY* *XIXYXZIZYX* *IIIZYXIYXZ* $d = 3$
11	*ZXYZXYIZZXX* *XZXZYIXXYZY* *YZXZIZYXXYY* *YYYIXZIIYZY* *ZZZZXZIXZYZ* *IZZXIIIXZXI* *ZZIXYZXZYXY* *IZZIYXIZXXY* *XYZZXZZXZIZ* *YIYYIIYIZXI* $d = 3$	*XIYZIZIIXZZ* *YZZYXIYXYYY* *IYIIZIYZYZY* *ZZIYZZYZIIZ* *IZYYZXXYYZX* *YYZZIYYXZYX* *ZZZZYZYXYIX* *ZXYIXXZYXYY* *IIZXIYYYYYY* $d = 3$	*ZIYIZXXXIZZ* *XZIYXZIIXZX* *IYZZIXIYXYX* *ZYIYIYZZXXI* *YXXZYIIXZYI* *YYZYYZXYYZX* *IYYXXXIZIXX* *YXXXYXIXIZY* $d = 3$
12	*IXIIZZIZIXYI* *XXIZXXIIIXII* *ZXXYIZYXIZYY* *ZYYYYIXZYXIX* *YZYYXYXZZXZI* *IZIZZYIXIZIZ* *ZZIXYYIZYYYZ* *YYIXZXZYIIZZ* *ZZXXXXZZXYYY* *ZZIIXXZIZIXI* *XYXZIYZXIXZZ* $d = 3$	*IIYXXIXZYIYI* *ZIXXZIZYIIIX* *YZZZZIYYIIZI* *IZXIYYZZXYII* *ZZYYIYZXYZXZ* *IYXIXYYYIYXX* *XYXYZXYIXYZZ* *XIXIYIZZZZYY* *XXYXXXXYXIIX* *YXZYXIIIZIXI* $d = 3$	*XYZYZIZIIXXX* *XXYIYIIZIIXZ* *YZZZYYYXIZII* *ZIIIXYXYXIYI* *XIZZZYYIXIZY* *IXXIIZXYXYXI* *ZZXIIYZZYIZI* *ZYZZIZXIIZZZ* *ZZXIYYZXIYXZ* $d = 3$

TABLE V. Generators and distances for the best weight-four codes found for the biased XZ channel using hill climbing.

$n\backslash k$	1	2	3
5	*ZIXZX* *IZXXZ* *ZYYIZ* *XZZIX* $d=3$	*XYYIX* *ZXIXX* *ZYYIZ* $d=1$	*XIYZZ* *IXYZZ* $d=1$
6	*IIXZZX* *YIIYXX* *ZZIIXY* *IXZIXX* *YIZIZY* $d=2$	*XIZXYI* *IYYXXI* *IZIZXX* *IZYIZY* $d=1$	*IXYYIX* *IIXZYX* *XXYIXI* $d=1$
7	*IZXIIZY* *ZZIXXII* *YIXZIZI* *IXIZXIZ* *IZZIIXY* *ZYYIZII* $d=3$	*IIYIYZX* *IZIXIYY* *ZYIIYIX* *XIIYXXI* *YIZXIIY* $d=2$	*YIZIIYY* *XIXYIIY* *IZYIXIZ* *IYIZIZX* $d=1$
8	*IZZIYIYI* *IIIYYIYY* *XXYIIIIY* *YIIXIXIX* *IIYZZYII* *YYIXXIII* *IIIIXZZY* $d=3$	*XIIZXIIX* *YIIYIXXI* *IIYXZYII* *XIIIZIYY* *IYXIIZYI* *ZXIIIYIZ* $d=2$	*IYIYIYYI* *IIIYXIYY* *XIIIYXXI* *YYXIIIYI* *IXYXIIIZ* $d=2$
9	*IXYYIXIII* *ZIIYZIIIY* *IIIIYIZYZ* *ZIYIIIYZI* *IIZYIZIYI* *IZIIIYIZY* *YIYIIIIXX* *IYIXYXIII* $d=3$	*YIXIIIXIX* *YYIIZXIII* *XIIIXIYYI* *IIXXIIXXI* *XIZIIYIYI* *IIYIZIIZY* *IXIYXIIYI* $d=2$	*XXXIXIIII* *YYIIIZIXI* *ZIYIIYXII* *YIIYZIYII* *IIYZYIIIY* *IIIYIYIYZ* $d=2$
10	*XIIXYIIXII* *IIYYZIXIII* *IIIXIIYIZY* *IIZIZIYYII* *ZIIIIXIYIY* *IZYIIZIXII* *IYIXIXYIII* *IZIIYIZIYI* *YZIYIIIIIZ* $d=3$	*IIIIZYIXIY* *XIIIIIYIXY* *IIIYZIZIYI* *IYXIIIXIZI* *XIYXIIYIII* *YIXIIXIIIX* *IXIIYXYIII* *YIIIXIXYII* $d=2$	*XIIIIYIYIX* *IXYXIIYIII* *XYXIYIIIII* *XYIYIIIIYI* *IIIYIXZIIY* *IIYIXYYIII* *ZIYIIIIXXI* $d=2$
11	*IIIIYZIZYII* *IZYYYIIIII* *YIIIYIIXXII* *IIIYIIZIIZY* *IIYIIIZIIYZ* *XIIIIIYIYXI* *YYIZIIIIIYI* *ZIIYIYIIZII* *IIXIXIIYIIY* *IYXIIIXIIYI* $d=3$	*IIIIIYZYIZI* *IZIYIIIYIIX* *IIIIYXIIZYI* *YXIIIIXXIII* *XIXIIIYIIYI* *XYIIIIIIZIY* *YXYXIIIIIII* *IXIXXIIIYII* *IIIIIIXIYXX* $d=2$	*IIYYIYIIZII* *YIIIXIYIIXI* *IYXZIIIIIIY* *XIIZYIIIYII* *YZIYIIIYIII* *XIIIIIIXIYY* *IIXIYXXIIII* *YIYIXIIIIIX* $d=2$
12	*IZIIIIYZYIII* *IIIIZYYIIZII* *IYXIIIIIZIIY* *XIIIIYIIYIIX* *IYIZXIZIIIII* *YXIXIIIIXIII* *IXZIIIXIIYII* *IIZIIIIYXIIY* *XIIYZIIIIIXI* *YIIIIZIIIYZI* *ZIIIIIIIYIYZ* $d=3$	*IIYXIIIYIIIZ* *IIXIZYIIIIIY* *ZIIYIIIIIYIY* *IIIIIIYIYYZI* *YIIZIIIZYIII* *IIIIYZXIIIYI* *YIIIIYZIIXII* *IIIIIIIXXIYY* *IYIIXXIIIYII* *IXYIXYIIIIII* $d=2$	*IZZIIIYIIIIY* *XIIXIZYIIIII* *YIIYYIIYIIII* *IIIYIYIIYIZI* *IYIIIIYYIIX* *IIYIIXXIIIYI* *IIIIXIIXZIYI* *YIZIIYIIIYII* *IXYIIIIIIZIY* $d=2$

TABLE VI. Generators and distances for the best weight-four codes found for the AD channel using hill climbing.

$n\backslash k$	1	2	3
5	*IXZXZ* *YZIYZ* *IZYZY* *ZXIZX* $d=3$	*ZXYIY* *ZXYYI* *XIYXX* $d=1$	*XYIXZ* *XYYIZ* $d=1$
6	*YIXXYI* *YXIIYZ* *ZIXIXZ* *IXIYZY* *XIZZXI* $d=1$	*YIYXXI* *IZYIXY* *YIZIYY* *ZXIYIZ* $d=2$	*IYYXYI* *XXYZII* *ZYIIYX* $d=1$
7	*ZYIIXXI* *ZIIYIXZ* *XIIXYIZ* *YZYZIII* *IYXZIIY* *IIZXYYI* $d=3$	*ZYIIXIX* *IXYIYYI* *ZIXZXII* *YXIYIYI* *IIIXZXY* $d=2$	*XYYIXII* *XIIXXXI* *ZIXYIIY* *ZZIIXYI* $d=1$
8	*ZIYIIYXI* *IZXYIZII* *YIIIZIZX* *IIIZIXYY* *YYIIIYZI* *IIXIXIZZ* *IIYYZYII* $d=3$	*XIYIIYYI* *IYIIIZXX* *IIXIXXIY* *IYYXYIII* *XXIZIIIY* *YIIYXIXI* $d=3$	*IXYXIIIZ* *YIIXYYII* *IIIZIXYX* *IYIIZZIY* *IZIIYIZY* $d=1$
9	*IIYIIXYYI* *IIIXXXIIX* *IYXYIYIII* *XYXIIIIXI* *IIXYIIXIY* *YIYIIXIIX* *IXYIXIYII* *XIIIYYXII* $d=3$	*XIXIIXIYI* *IIIYIZXXI* *IIYIXIIXY* *YIIXYIIZI* *YYIIIYYII* *IXIXIIZIY* *IXXIIIXIX* $d=2$	*YIYIIIYIZ* *XXIIIXIIX* *IXXIZIXII* *XIZIXIIXI* *YIIXIYIZI* *IYIIXIXIY* $d=1$
10	*YIIYYIIIIY* *IIIXIXIXIX* *IYYIIIIYIY* *XXXIXIIIII* *IYIIYIYIZI* *IIIXXXIIXI* *ZIYIXIXIII* *IIIIIZXIYY* *XIXXIIIXII* $d=3$	*IIYXIYYIII* *IIIIYXZYII* *IXZIXIXIII* *IYYIIIIIXY* *YZIIYIIIIX* *IIIYIXIIZX* *IIIZIIXXXI* *XIIIXYIIXI* $d=3$	*IIIIZXXIXI* *YYXIIIIIXI* *XIYZIIIIIX* *YIIXIXIYII* *IIIIYYIXIX* *IIZIXIIIYY* *IIIYIZYIIY* $d=1$
11	*YIYIIYIIIIY* *IIXIYZIYIII* *IXIYIIZYIII* *IXYIXIIIIXI* *IIZIIXIIYYI* *IIIXIYIXIXI* *XIIXIIXIIIX* *ZXIIIIIIYIZ* *YIIIIZIXXI* *IYZIIIIXIIX* $d=3$	*IIIIIIXIYXY* *IIIZIIYXIIX* *IYIYIXIZIII* *IYZIIIIIXYI* *YIXIYIIIIYI* *IXIIYIIXYII* *ZIYIIXIIIXI* *IIIIXXYIXII* *YIIXIZXIIII* $d=3$	*YIIIIIXIXIY* *XIIIIYIXZII* *IIXYXIIIIIY* *IYIIXIIXIZI* *IIXIIZIYIXI* *IXIXZIXIIII* *IIZXIXIIXII* *XZIIIIYIIXI* $d=1$
12	*IXIIYIIXXIII* *IIXIIIIXIYIX* *IIIIXIIIYXIY* *IIIIXYIYIIIY* *IIIYYXIIIYII* *ZXIIIIZIIIXI* *IZIIIIYIZYII* *IIYXIYIIIIIY* *YIIXIYIIIIYI* *YIIIIIXIXYII* *XIXYIIYIIIII* $d=3$	*IIYZZIIIIIYI* *IYIIIZIIIYYI* *IIXIYIIYIIIX* *IIIIXYIIIIZY* *YIIIIXIIIXIX* *YIYIIIYXIIII* *IXIYIIIIXIZI* *ZXIIIYXIIIII* *IYIIIIZYZIII* *IIIIIIIXXYIY* $d=3$	*IIIYXZIIXIII* *XIIYIIIIIXX* *IIYZIXIIIIIY* *IIIIZIIIYYXI* *ZIIIIIYXIIIY* *YIIIIIIYIZYI* *IYIIIIIXXXII* *IXYIIIZYIIII* *IZZIIYIIIYIII* $d=3$

TABLE VII. Generators and distances for the best performing inequivalent cyclic codes with weight-four generators on the biased XZ and AD channels. If a code requires two generators, they are grouped in brackets; otherwise, a single generator is given as in Table II. The generators of codes performing well on both channel types are given in bold, while the generators for codes previously appearing in Table II are marked with an asterisk.

$[[n,k]]$	Biased XZ	AD
[[5,1]]	***YZIZY***, $d=3$*	***YZIZY***, $d=3$*
[[6,1]]	*YIZZIY*, $d=2$*	*YZIIZY, $d=2$* *XZIIZX, $d=2$*
[[6,2]]	*YZIZYI*, $d=2$*	*XIZIXY*, $d=2$* *YIZIYX*, $d=2$*
[[6,3]]	***XIYXIY***, $d=2$* ***YZIYZI***, $d=2$*	***XIYXIY**, $d=2$* ***YZIYZI**, $d=2$* *XZIXZI, $d=2$*
[[7,1]]	***XZIZXII***, $d=3$*	***XZIZXII***, $d=3$* *YZIZYII*, $d=3$*
[[8,1]]	***ZZYIIIIY***, $d=3$*	***ZZYIIIIY**, $d=3$* *ZZXIIIIX, $d=3$*
[[8,2]]	***YIIXIIYX***, $d=2$* *YIIZIIYZ*, $d=2$*	***YIIXIIYX***, $d=2$* *XIIYIIXY*, $d=2$*
[[8,3]]	***YIIIIYYY**, $d=1$* ***XIIIIXXX**, $d=1$*	***YIIIIYYY**, $d=1$* ***XIIIIXXX**, $d=1$*
[[9,1]]	***ZIZYIIIIY***, $d=3$*	***ZIZYIIIIY***, $d=3$* *ZIZXIIIIX*, $d=3$*
[[9,3]]	***IIIYYIYYI**, $d=1$* ***IIIXXIXXI**, $d=1$*	***IIIYYIYYI**, $d=1$* ***IIIXXIXXI**, $d=1$*
[[10,1]]	*IIYZIIIIZY, $d=2$*	*XIIZIIZIIX, $d=3$* *YIIZIIZIIY, $d=3$*
[[10,2]]	***IYXIIIIIXY**, $d=3$* *IZYIIIIIYZ, $d=3$*	***IYXIIIIIXY***, $d=3$*
[[11,1]]	***IYIIZIIZIIY***, $d=3$*	***IYIIZIIZIIY***, $d=3$* *IXIIZIIZIIX*, $d=3$*
[[12,1]]	***IIIIYIIZZIIY**, $d=3$*	***IIIIYIIZZIIY**, $d=3$* *IIIIXIIZZIIX, $d=3$*
[[12,2]]	*YIIIIZIIIIYZ, $d=2$*	*XZIIIIIIIZXI, $d=3$* *YZIIIIIIIZYI, $d=3$*
[[12,3]]	*(**XIIXIIXIIXII**,* ***IYIIIIYIIYYI**), $d=2$*	*(**XIIXIIXIIXII**,* ***IYIIIIYIIYYI**), $d=2$* *(YIIYIIYIIYII,* *IXIIIIXIIXXI), $d=2$*

F. CSS codes

We next consider CSS codes, which, as outlined in Sec. II D, are codes that can be represented using generators that contain either only X or only Z matrices as their nonidentity elements. Similar to the search for weight-four codes, we must modify both the initial stabilizer construction and the generator permutation. In particular, when adding a new generator, we will select a suitable X-only element half the time and a Z-only element the other half. Another required modification is the removal of the permutation mutation as, in general, it does not map CSS codes to CSS codes. We also consider cyclic CSS codes, which can be thought of in two equivalent ways. They can be viewed as codes with a binary representation where $\tilde{H}_X$ and $\tilde{H}_Z$ each correspond to a binary cyclic code. Alternatively, they can be considered in the GF(4) framework as additive cyclic codes that can be represented by a 1-only cyclic generator and/or an ω-only cyclic generator. The number of these cyclic CSS codes is given in the fourth column of Table I. We also consider the family of dual-containing CSS codes to generalize the result of Ref. [4], where it was shown that the [[7,1,3]] Steane code [39], which has

$$\tilde{H}_X = \tilde{H}_Z = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{61}$$

performs poorly on the biased XZ channel. We have constructed these codes by enumerating all of the inequivalent binary self-orthogonal codes using SAGEMATH [40] (recall that a generator matrix for a binary-self orthogonal code is the parity-check matrix for a dual-containing code). The number of such codes is given in the fifth column of Table I. Note that there can only be an $[[n,k]]$ dual-containing CSS code if $n-k$ is even; furthermore, even when $n-k$ is even, not many of them exist for the parameters considered.

As can be seen for the biased XZ channel in Fig. 10, both the CSS codes found via hill climbing and the cyclic CSS codes perform poorly compared to their non-CSS counterparts. This performance can be improved by following the modification outlined in Ref. [5], which involves applying the permutation $Z \leftrightarrow Y$ to the code's generators (this is motivated by the fact that Z-only generators commute with any Z-only error, meaning that they often provide no information about an error when η is large). Given the nature of this modification, we call such codes CSSY codes. We have performed a hill-climbing search for CSSY codes, and it can be seen that they perform significantly better than the standard CSS codes; however, they are still outperformed by non-CSS codes in most instances. Similarly, while the cyclic CSSY codes perform better than the cyclic CSS codes, there is often a significant performance gap to the non-CSS cyclic codes. The dual-containing CSS codes perform poorly across the board, which can at least partially be attributed to the fact that they must have $d_X = d_Z$. Furthermore, their performance cannot be improved as they are invariant under a $Z \leftrightarrow Y$ permutation. As shown in Fig. 11, the results on the AD channel are similar to those for the biased XZ channel. Both the CSS codes found via hill climbing and the cyclic CSS codes perform poorly compared to the non-CSS codes. In this case, the performance gain of the CSSY codes over the CSS codes is less pronounced. A notable exception to this is the [[9,1]] case where the best CSSY code found via hill climbing outperforms the best unrestricted code found. Somewhat surprisingly, after applying an $X \leftrightarrow Y$ permutation to the second, fourth, fifth, sixth, and ninth qubits, this code is equivalent to the best code with weight-four generators found in Sec. IV E. Again, the performance of the dual-containing CSS codes is very poor compared to nearly all other codes considered. Generators for the best CSSY codes found via hill climbing can be found in Tables VIII and IX. We omit the standard CSS codes found via hill climbing and the cyclic CSS(Y) codes due to their poor performance.

TABLE VIII. Generators and distances for the best CSSY codes found for the biased XZ channel using hill climbing.

$n\backslash k$	1	2	3
5	*XXXXX* *IIIYY* *IYIYI* *IIYIY* $d=1$	*XXXIX* *YYIYI* *YIYYI* $d=1$	*XXXXI* *XXIXX* $d=1$
6	*XXXXXX* *IIYYYY* *IYYIYY* *YIYYIY* *IYYYYI* $d=2$	*IXXXXX* *YIYYII* *YIYIYI* *YYIYII* $d=1$	*IXIXXX* *YYIIYI* *YIYIYY* $d=1$
7	*IIIXIXX* *XXXXXIX* *YYIYYYI* *YIIYYIY* *IYIIYII* *YYYIIYY* $d=2$	*XXXXXXX* *YYIIYYI* *YYIYIYI* *IYYYYYY* *YYIYIIY* $d=2$	*XXXXXXX* *YIIYYYI* *YYIIYIY* *YYYYIII* $d=2$
8	*XIXXIIXX* *IXIXXXXI* *IYYIIYIY* *IYYIIIYI* *YIYIYIYY* *IYYYIYYY* *YYYYIIIY* $d=2$	*XXXXIIXX* *XIXXXXXI* *IIIIIYYY* *IYYYIYYI* *YYIIIYII* *IYYIYYYY* $d=2$	*XXIXXXXX* *XXXXIXXX* *YIIIIYYY* *YYIYIIYI* *IIYYYIYY* $d=2$
9	*IIXXIIIXX* *XXIXXIIXX* *IXXXXIXXI* *IIIXXXIIX* *IIYYYYIYY* *IYYIIYIIY* *IYIYYYYIY* *YIIIYYYII* $d=3$	*IXXXXIXIX* *XXIIXXXXI* *XIXIXXXII* *YIIYYIIII* *YYIIIYIYY* *IIYYIYIYI* *YIIYYYYIY* $d=2$	*XXIIXXXIX* *IIXXXXIXX* *YYIYYIYII* *IYYYIYYIY* *IIIYYIIYY* *IIIYIYYII* $d=2$
10	*IXXIXXXIXI* *IIXXIIXIXX* *IXIXIIXXXI* *XIIIXIIXXI* *YYIIIIIIYY* *YIYYIIIIYY* *IYYIIIYYYY* *YIYIIYYIYY* *IIIYYIIIYI* $d=3$	*IXIXIIXXXX* *IXXXXXXIIX* *XIIXXIXXII* *YIIYIYYYIY* *YIYYIYIIIY* *IIYIYIYIIY* *YYYIYIIIIY* *IYIIIYIIYI* $d=2$	*XIIXXXIXIX* *IIXIXXXIXI* *XXIIXIIXXI* *YYYYIIYIYI* *YIIIIIYIYY* *YIYYYYYIII* *IIIIYYIYIY* $d=2$
11	*XIXXIIXIXIX* *IXIIXIXXIXX* *IIIXXXXIXXI* *XXIIXIIXXIX* *IIIIIIYYYII* *YYIIYIYYYYY* *IYYIIIIIIIY* *IYYYYIYYIYY* *YYYIYIYIYYI* *IIYIYYIIIIY* $d=3$	*IIXXIXXIXXX* *XIIXIIXXXII* *IIXIXIXIXXI* *IXXXIIXXIIX* *YIYYIIIYYIY* *YYYIIYYIIIY* *YYIYYIIIIYI* *YIYIIIYIIYI* *IYYIYIYIYIY* $d=3$	*IXIIXXXIIIX* *XIIXXIXXIXX* *XIIIIXXXXIX* *XXXIIXXIXXI* *YIYYIYYYIII* *IIYYIYIYYYY* *YYYIYIIIYYI* *YIIIYYIYYYI* $d=3$
12	*XXXIIXIIXXXX* *IXIXXXXXIXXI* *IIXXIIXXXXII* *XXIXXXXIIIII* *XIIXIXXXXXXI* *IIXXIIIIIXIX* *IIYIYIYYYIYY* *YYIIYIYIYIYI* *YIIIYYYYIIII* *IIYYIIYYIIYI* *IIYIYYIYYYII* $d=3$	*XXIXIIXIXXIX* *XIXIIIXXXIXI* *XXXIIIXXIXXX* *IIIXXXIXXIIX* *IIYIIIYYIIYY* *IYYIYIYIYYYI* *IYYYIYIYYYYI* *IYIIIIYIIYY* *IIYIIYIIIYYY* *YYYYYIIYIIYY* $d=2$	*XIIXIXXIXXII* *IIXXXIXXXIIX* *XXXIIIIXXIII* *XXXIXIXIIXXI* *IYYYYYYYYIII* *YIYIIIYYYYII* *YIIYIYYYIIIY* *YIYYIIIIIIYY* *YIYIIIYIIIYI* $d=3$

TABLE IX. Generators and distances for the best CSSY codes found for the AD channel using hill climbing.

$n\backslash k$	1	2	3
5	*IIXIX* *IIIXX* *IYYYY* *YIYYY* $d=1$	*XIXXI* *XXXII* *IYYYY* $d=1$	*IIYYI* *YYYIY* $d=1$
6	*XXXXXI* *XXXXIX* *YIIYII* *IYIYII* *YIYYYY* $d=2$	*XXXXXX* *IXIXXI* *IYYYIY* *YIIYYY* $d=2$	*IXXIXI* *YIYYYI* *YYIIYY* $d=1$
7	*XXXXIII* *XXIIXXI* *XIXIIXX* *IYIYIYY* *YYIIYYI* *IIYYYYI* $d=3$	*IXIIIIX* *XXXIXXI* *XXXXIIX* *IYYYYYY* *YYYIYIY* $d=2$	*XXXXXXX* *IIYYYIY* *YYIIYIY* *IYYIIYY* $d=2$
8	*IXIXXIII* *XXXXXXXI* *IXIXIXII* *XXIIXXXX* *YYIIYYII* *YYYIYYYY* *IYIYIIIY* $d=2$	*IXXIIIIX* *XXXIXIIX* *XXIXIXXX* *YYIIYYIY* *YYIYYIIY* *IIYYIYYY* $d=2$	*XIXIXXII* *IXXIIXXX* *XIIXIXXX* *YYYYYYIY* *IIIIIIYY* $d=2$
9	*IIXXXIXXX* *XXXIIIXII* *XIXIXXXIX* *IXIXIIXIX* *YIIIIYYIY* *IYYYIYIII* *YIYIYYIII* *YIIYYYYYI* $d=3$	*IXXXXXIIX* *XXIXIIIXI* *IIXXIIXXX* *XIXXIXXXI* *YYYIYIIIY* *YYIYYYIYI* *IYIYIIYII* $d=2$	*IXXXIXIXX* *IIIXXIXXX* *XIXIXXIXI* *YYYIIYYYI* *YYIYIIYYY* *IIIYYYIII* $d=2$
10	*XIIXIXIXXX* *IXXXXIIIIX* *XIXXXIXIXX* *IXXIIIXXXX* *YYYIIYIIII* *IYIYYYYYIY* *IIYIIIYYY* *IIYIYIYYYI* *YIYIYYYIII* $d=3$	*XXXIXXIIXI* *XIXXIIXXXI* *IXXIIXIXXX* *IXXXIIIXII* *YYIIYYIYIY* *YYYIIIYIYY* *YYIYYIYIYI* *IYIYYYIIYY* $d=3$	*XXIXIXXIIX* *XXXIXXIIXI* *IIIIIXIXXX* *XIIXXXIIII* *IYIIYYYYYY* *YIIYYYYIYI* *IIYYIYYIIY* $d=2$
11	*XIXIIXIXIII* *XIIXIXIIIXI* *IXXXIIIIXIX* *XXXIXXXXIIX* *IXIIXXIIXXI* *IYIIYIYIIIY* *IIIYYIIIIYY* *IIYYYYYIIII* *YYIYIYIIYYY* *YYIIIIIYIYY* $d=3$	*XIXXXIIXXXI* *XXXIIXXIXIX* *IIXIIXIXXXX* *XIXXIIXXIIX* *YIIIIYYIYII* *IIYIYYIYIYI* *IYIYIIYYYYY* *IIIYYYYYYIY* *YIYYYYIYYII* $d=3$	*XXXIIXXXIXI* *IXXXXIXXIII* *IXIIXXXIXXX* *IXXIIXIIIIX* *YIYYIYIIIYI* *YIYYIIIYYIY* *YYYIYYIYIYY* *YIIYYIYYYYI* $d=3$
12	*XIIXIIXXXIII* *XXIXIIIIXIIX* *XXIIXXXXIXIX* *IIXIXIXXXIXX* *IIXXIXIXXXX* *IIXIXIIIIXIX* *IIIXXIXIXXXX* *YYYIIIYIIYII* *IYIYYYYYYIYY* *YYYYIIIYYYYI* *YIYYIYYIYIIY* *IYYYYYYIIIYI* $d=3$	*XXXXIIIIIIXI* *XXXIXIIIIXIX* *XIIXXXXIIXII* *XXIXXXIIXXXX* *XXXXXIIXXIIX* *IIYIYIIYIYYY* *YYIIIIIYIYIY* *IIIYYYIYYYYI* *IYIYYIYYYYIY* *YYYYIIYIIYII* $d=3$	*XXIIIIIXXXXI* *XIXIXXIIXIXI* *XXIIXXXXXIIX* *XXIXIXXIIIXI* *YYIYIIIYIIYY* *IYIYYIIYYYII* *IIYYIIIIIYYI* *IYIIYYIIIYIY* *YYYIYIYIIYYI* $d=3$

TABLE X. Generators and distances for the best linear codes found for the biased XZ channel using hill climbing.

$n\backslash k$	1	2	3
5	*XXYIY* *ZZXIX* *XYIYX* *ZXIXZ* $d=3$	–	–
6	–	*YIXIYX* *XIZIXZ* *XYZZZY* *ZXYYYX* $d=2$	–
7	*ZIZXZZY* *YIYZYYX* *IYYYZII* *IXXXYII* *ZYYIIXI* *YXXIIZI* $d=3$	–	*IXZYXXZ* *IZYXZZY* *YYXZXIY* *XXZYZIX* $d=2$
8	–	*IYXZYYXI* *IXZYXXZI* *XIYXYYIY* *ZIXZXXIX* *IXYYYXIY* *IZXXXZIX* $d=3$	–
9	*ZZIIIXIIY* *YYIIIZIIX* *IXZZZXZZX* *IZYYYZYYZ* *IYIZZIZXX* *IXIYYIYZZ* *IIYYIIXXI* *IIXXIIZZI* $d=3$	–	*ZXXXYXXIZ* *YZZZXZZIY* *ZZZXIYXZY* *YYYZIXZYX* *ZYZIIZZXI* *YXYIIYYZI* $d=3$
10	–	*XZYXXXYZZX* *ZYXZZZXYYZ* *ZYXYXZZXXX* *YXZXZYYZZZ* *YXIZYYXIYZ* *XZIYXXZIXY* *ZIYYYXYZYI* *YIXXXZXYXI* $d=3$	–
11	*YZXIIYYZIZX* *XYZIIXXYIYZ* *YYXZYXIZYYY* *XXZYXZIYXXX* *YIXYYXZXZZY* *XIZXXZYZYYX* *IZZYXIYIYXX* *IYYXZIXIXZZ* *ZXZXIZIIZXZ* *YZYZIYIIYZY* $d=3$	–	*XIZZZIYYIXY* *ZIYYYIXXIZX* *XYXIYIYXYIZ* *ZXZIXIXZXIY* *XXZXXIIZYYI* *ZZYZZIIYXXI* *ZIYIYYYYIYY* *YIXIXXXXIXX* $d=3$
12	–	*IIIIYYIIIIZZ* *IIIIXXIIIIYY* *YYZYZYIIYIIY* *XXYXYXIIXIIX* *IZZZZZZIXXYY* *IYYYYYYIZZXX* *ZXYIYYXYIZII* *YZXIXXZXIYII* *ZYYXZZIIZXZZ* *YXXZYYIIYZYY* $d=4$	–

TABLE XI. Generators and distances for the best linear codes found for the AD channel using hill climbing.

$n\backslash k$	1	2	3
5	*XIXZZ* *ZIZYY* *IXZXZ* *IZYZY* $d=3$	–	–
6	–	*IXIXYY* *IZIZXX* *YXXIIY* *XZZIIX* $d=2$	–
7	*IXYYXYZ* *IZXXZXY* *XXIYYYY* *ZZIXXXX* *XZXYIZY* *ZYZXIYX* $d=3$	–	*XZZZIXY* *ZYYYIZX* *IXYZYXY* *IZXYXZX* $d=2$
8	–	*IYIYZYXX* *IXIXYXZZ* *ZIIXYXYX* *YIIZXZXZ* *ZZZXXIIZ* *YYYZZIIY* $d=3$	–
9	*YXXIXZXZY* *XZZIZYZYX* *ZXYYXYIZZ* *YZXXZXIYY* *IZYYIYXIX* *IYXXIXZIZ* *IIIIYIZYZ* *IIIIXIYXY* $d=3$	–	*ZZIXIZIXY* *YYIZIYIZX* *YZIXXXXYX* *XYIZZZZXZ* *IXYXYXXZZ* *IZXZXZZYY* $d=3$
10	–	*XYYIIYZZYX* *ZXXIIXYYXZ* *ZXYIYIXXZZ* *YZXIXIZZYY* *XXIIYZZXYY* *ZZIIXYYZXX* *YXXYYXXIIZ* *XZZXXZZIIY* $d=3$	–
11	*YIYZYZZYXXY* *XIXYXYYXZZX* *YXIIIYZZIXI* *XZIIIXYYIZI* *XYXZIZYIXIZ* *ZXZYIYXIZIY* *XIZXIXXXZIX* *ZIYZIZZZYIZ* *YXXZXYYIXYX* *XZZYZXXIZXZ* $d=3$	–	*XZXYYIIXIZZ* *ZYZXXIIZIYY* *YIIXIZYXXII* *XIIZIYXZZII* *ZIIIXZZYZZY* *YIIIZYYXYYX* *IYXZZZIZXIZ* *IXZYYYIYZIY* $d=3$
12	–	*IIIIYXYYXZXX* *IIIIXZXXZYZZ* *IIZIXZZZIZZY* *IIYIZYYYIYYX* *IZIZIXYIXZII* *IYIYIZXIZYII* *ZIYXIXXIIIYI* *YIXZIZZIIIXI* *XIYIYIXXIYZX* *ZIXIXIZZIXYZ* $d=4$	–

G. Linear codes

The dual-containing CSS codes considered in the previous section are examples of linear stabilizer codes. An additive $(n, 2^{n-k})_4$ code $\mathcal{C}$ is linear if and only if it has a generating set of the form $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{(n-k)/2}, \omega\boldsymbol{b}_1, \ldots, \omega\boldsymbol{b}_{(n-k)/2}\}$. This corresponds to the stabilizer having generators of the form $\mathcal{S} = \langle M_1, \ldots, M_{(n-k)/2}, \bar{M}_1, \ldots, \bar{M}_{(n-k)/2}\rangle$, where $\bar{M}_i$ is a version of M_i that has been subjected to the permutation $(X, Y, Z) \to (Z, X, Y)$. To search for such codes, we must first modify the initial construction and generator mutations. In particular, we add or remove the generators M_i and $\bar{M}_i$ in pairs. To preserve linearity, the permutation mutation has to be restricted to permutations corresponding to a multiplication of a coordinate of $\mathcal{C}$ by ω or $\bar{\omega}$. That is, the permutation must either be $(X, Y, Z) \to (Z, X, Y)$ or $(X, Y, Z) \to (Y, Z, X)$. We also consider linear cyclic codes, the structure of which is outlined in Sec. II B. The number of such codes is given in the sixth column of Table I. Like the dual-containing CSS codes, $[[n, k]]$ linear codes can only exist for even $n - k$; furthermore, while $n - k$ is even for [[5,3]] codes, there are no linear codes with these parameters that involve every qubit.

As shown in Fig. 10, the linear codes found via hill climbing perform reasonably well on the biased XZ channel. The performance of the linear cyclic codes is somewhat less impressive, with there being a significant gap in performance to the more general additive cyclic codes. This can potentially be attributed to the fact that, at least for the code parameters considered, there are very few linear codes. As can be seen in Fig. 11, the linear codes found via hill climbing for the AD channel perform better than those on the biased XZ channel, particularly in the $k = 3$ case. However, the linear cyclic codes still perform poorly. The best linear codes found via hill climbing are given in Tables X and XI. We omit the linear cyclic codes due to their poor performance.

V. CONCLUSION

We have shown that the error rate of an optimal stabilizer code decoder can be effectively approximated by considering only a limited subset $\mathcal{E}$ of the 4^n possible Pauli errors, and we have outlined how to construct $\mathcal{E}$ without having to enumerate all of these errors. Utilizing this approximate calculation, we have demonstrated that there are a number of $[[5 \leqslant n \leqslant 12, 1 \leqslant k \leqslant 3]]$ cyclic stabilizer codes that perform very well on both the biased XZ and AD channels across a range of error probabilities and biases. We have also shown that an indication of the performance of a stabilizer code can be obtained by considering the error rate of an associated $[2n, n + k]$ classical code. We have used this as the basis for a hill-climbing algorithm, which we have shown to be effective at optimizing codes for both of the asymmetric channels considered. Furthermore, we have demonstrated that by modifying the mutation operation of this hill-climbing algorithm, it is possible to search for highly performant codes that satisfy structure constraints. In particular, we have successfully performed searches for codes with weight-four generators, CSS(Y) codes, and linear codes.

[1] D. E. Gottesman, Ph.D. thesis, California Institute of Technology, 1997; arXiv:quant-ph/9705052.

[2] Z. W. E. Evans, A. M. Stephens, J. H. Cole, and L. C. L. Hollenberg, arXiv:0709.3875.

[3] L. Ioffe and M. Mézard, Phys. Rev. A **75**, 032345 (2007).

[4] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, Phys. Rev. Appl. **8**, 064004 (2017).

[5] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, Phys. Rev. Lett. **120**, 050505 (2018).

[6] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, Proc. R. Soc. London, Ser. A **465**, 1645 (2009).

[7] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).

[8] A. Steane, Proc. R. Soc. London, Ser. A **452**, 2551 (1996).

[9] S. A. Aly, in *2008 International Conference on Computer Engineering and Systems, Cairo, Egypt* (IEEE, Piscataway, NJ, 2008), pp. 157–162.

[10] M. F. Ezerman, S. Jitman, S. Ling, and D. V. Pasechnik, IEEE Trans. Inf. Theory **59**, 6732 (2013).

[11] L. Wang, K. Feng, S. Ling, and C. Xing, IEEE Trans. Inf. Theory **56**, 2938 (2010).

[12] G. G. La Guardia, Quant. Info. Proc. **12**, 2771 (2013).

[13] K. Guenda and T. A. Gulliver, Int. J. Quantum. Inform. **11**, 1350047 (2013).

[14] P. Iyer and D. Poulin, IEEE Trans. Inf. Theory **61**, 5209 (2015).

[15] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, Phys. Rev. Lett. **78**, 405 (1997).

[16] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, IEEE Trans. Inf. Theory **44**, 1369 (1998).

[17] W. C. Huffman, Adv. Math. Commun. **1**, 427 (2007).

[18] W. C. Huffman, Adv. Math. Commun. **2**, 309 (2008).

[19] E. Berlekamp, R. McEliece, and H. Van Tilborg, IEEE Trans. Inf. Theory **24**, 384 (1978).

[20] A. Rigby, J. C. Olivier, and P. Jarvis, Phys. Rev. A **100**, 012330 (2019).

[21] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes* (Elsevier, Amsterdam, Netherlands, 1977).

[22] A. Luo, Master's thesis, Concordia University, Montreal, Quebec, Canada, 2004.

[23] K. Kraus, *States, Effects, and Operations: Fundamental Notions of Quantum Theory* (Springer, Berlin, 1983).

[24] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2011).

[25] J. Napp and J. Preskill, Quantum Inf. Comput. **13**, 490 (2013).

[26] D. P. DiVincenzo, D. W. Leung, and B. M. Terhal, IEEE Trans. Inf. Theory **48**, 580 (2002).

[27] J. Emerson, R. Alicki, and K. Życzkowski, J. Opt. B: Quantum Semiclassical Opt. **7**, S347 (2005).

[28] C. Dankert, R. Cleve, J. Emerson, and E. Livine, Phys. Rev. A **80**, 012304 (2009).

[29] M.-H. Hsieh and F. Le Gall, Phys. Rev. A **83**, 052331 (2011).
[30] K.-Y. Kuo and C.-C. Lu, in *2012 International Symposium on Information Theory and Its Applications, Honolulu, HI* (IEEE, Piscataway, NJ, 2012), pp. 208–211.
[31] K.-Y. Kuo and C.-C. Lu, arXiv:1306.5173.
[32] F. Gaitan, *Quantum Error Correction and Fault-Tolerant Quantum Computing* (CRC Press, Boca Raton, FL, 2008).
[33] I. Djordjevic, *Quantum Information Processing and Quantum Error Correction: An Engineering Approach* (Elsevier, Oxford, UK, 2012).
[34] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, IEEE Trans. Inf. Theory **50**, 2315 (2004).
[35] A. M. Steane, Phys. Rev. A **54**, 4741 (1996).
[36] R. A. Brualdi, *Introductory Combinatorics*, 5th ed. (Pearson Education Asia, Beijing, China, 2009).
[37] G. E. Andrews and K. Eriksson, *Integer Partitions* (Cambridge University Press, Cambridge, UK, 2004).
[38] S. Luke, *Essentials of Metaheuristics* (Lulu, Morrisville, NC, 2013).
[39] A. M. Steane, Phys. Rev. Lett. **77**, 793 (1996).
[40] SageMath, the Sage Mathematics Software System (Version 8.7), The Sage Developers, 2019, https://www.sagemath.org.